UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS

Facultad 3

CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES

Departamento de Soluciones Empresariales

WAOX v1.0: DASHBOARD PARA EL MONITOREO DE INFORMACIÓN EN TIEMPO REAL

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

AUTOR: Andy Pérez Herrera.

TUTOR: MsC. Sasha Valdés Jiménez.

COTUTOR(ES): Ing. Omar Antonio Díaz Peña.
Ing. Javier Ruiz Durán.

Ciudad de La Habana, 2012

DECLARACIÓN DE AUTORÍA

Declaro que soy el autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2012.

AUTOR

TUTOR
MsC. SASHA VALDÉS JIMÉNEZ

CO-TUTOR
ING. OMAR ANTONIO DÍAZ PEÑA

CO-TUTOR
ING. JAVIER RUIZ DURÁN

AGRADECIMIENTOS

En especial a mis padres por guiarme siempre por el buen camino y confiar en mí.

A mi hermano Andrew, por estar siempre presente.

A mi novia Dailén por apoyarme en todo momento.

A Analina, Reinier, Osmin, Eduardo, Pablito, Gabriel, Raúl, Lino, José Manuel, Rafael, Yosmany, Felipe y Pedro Frank por su amistad incondicional.

A todos mis amigos del aula, la residencia y las canchas.

A Collado por ayudarme incondicionalmente.

A mis tutores por guiarme durante el desarrollo de la investigación.

A todos los que de una forma u otra hicieron posible la realización de este trabajo.

A todos los profesores que de una forma u otra han contribuido en mi formación.

DEDICATORIA

Este trabajo está especialmente dedicado a mi madre Daelé Herrera Díaz por guiarnos y educarnos tan bien.

A mi papa Andrés Pérez Díaz por ser mi ídolo y ejemplo a seguir, por sacrificarse junto a mi madre y guiar a sus dos hijos por el mejor camino.

A mi novia Dailén Gutiérrez Alfonzo por darme su amor incondicional y siempre estar a mi lado.

A toda mi familia por apoyarme siempre.

RESUMEN

El marco de trabajo Sauxe, desarrollado por el Centro de Informatización de la Gestión de Entidades (CEIGE), provee una estructura genérica para el desarrollo de aplicaciones web de gestión posibilitándole al desarrollador lograr una mayor estandarización, flexibilidad, integración y agilidad en el desarrollo del producto final.

El marco de trabajo Sauxe no permite mostrar en tiempo real la información generada por los procesos de análisis de información de las aplicaciones desarrolladas sobre él, con lo que se dificulta la toma de decisiones de sistemas tan complejos dedicados a la planificación de recursos empresariales y estadísticos, por citar algunos.

Este trabajo tiene como objetivo el desarrollo de una herramienta que permita monitorizar en tiempo real la información generada por las aplicaciones desarrolladas con el marco de trabajo Sauxe. De esta manera se persigue facilitar a toma de decisiones, ahorrar tiempo en comparación a la ejecución de varios informes, identificar de una manera rápida los valores extremos, creando informaciones visuales y gráficas en tiempo real. El desarrollo fue basado sobre políticas de software libre empleando las tecnologías Apache, PostgreSQL, PHP, XMPP, JavaScript.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad, se ha vuelto indispensable el uso de las Tecnologías de la Información y las Comunicaciones (TIC). El constante progreso y la gran aceptación de estas tecnologías se deben a los múltiples y convenientes beneficios que brindan. Debido al gran volumen de datos que estas generan, el diseño de nuevas herramientas para el tratamiento de la información que tributen a la toma de decisiones en todos los niveles, se ha convertido en un objetivo primordial para la gestión de la información.

La gestión empresarial, encargada de la *planificación, organización, dirección y control* de los recursos de una organización con el fin de obtener el máximo beneficio posible, está potenciada por una gran cantidad de herramientas informáticas que facilitan el logro de sus objetivos. El software de gestión empresarial ha surgido para impulsar la incorporación de las TIC en los procesos de trabajo de las empresas posibilitando la toma de decisiones rápida y segura, acortando los ciclos productivos e incrementando la calidad de los servicios y productos.

Los sistemas de gestión empresarial, como los de Administración de Relaciones con los Clientes (CRM[1]) y los de Planificación de Recursos Empresariales (ERP[2]), facilitan las estrategias de negocio de todo tipo de empresas. CRM implementa un modelo de negocios cuya estrategia está destinada a lograr identificar y mantener las relaciones con los clientes manteniendo su continuo valor agregado. Los ERP son soluciones informáticas cuyo objetivo es gestionar la información a través de las diferentes áreas para agilizar las tareas, mejorar los procesos de producción y reducir costos. (1)

En el entorno competitivo actual, el análisis de información para predecir las tendencias del mercado y para mejorar el rendimiento de la empresa es una actividad esencial del negocio. Sin embargo, es cada vez más claro que el éxito empresarial requiere de un análisis de datos y respuestas inmediatas con el fin de cumplir con los rápidos cambios en la demanda de los clientes.

Las herramientas de análisis y reportes surgen a principios de los años 90, creándose la tendencia de generar reportes personalizados, sin depender exclusivamente de los departamentos de sistemas. La combinación de estas herramientas con las bases de datos, clamaron el arribo de la era del “auto-servicio” por los proveedores de software para la Inteligencia de Negocios. (2)

[1] *Customer Relationship Management.*
[2] *Enterprise Resource Planning.*

El contexto que enfrentaron muchos de los usuarios se caracterizó por un entorno con herramientas muy complejas, donde un solo reporte o varios presentados de forma desagregada no contribuían con el manejo de la información de una manera eficiente para la toma de decisiones estratégicas.

Como solución, aparecieron las herramientas informáticas conocidas como **dashboard**. Esta herramienta muestra la información más importante, presentado en una manera que permite hacer un seguimiento de lo que está ocurriendo en un instante. La mayoría de los dashboard que se utilizan en los negocios de hoy están muy por debajo de su potencial. Para servir a su propósito y maximizar sus prestaciones, los dashboard deben mostrar una densa red de información en una pequeña cantidad de espacio de manera que se comunique con claridad e inmediatez. Esto requiere un diseño que se nutra y aproveche el poder de la percepción visual para lograr el procesamiento de grandes cúmulos de información. (3)

El uso de los dashboard unido a los ERP o CRM crea una ventaja visual, que facilita el acceso, extracción y análisis rápido a los datos. Los mismos otorgan una nueva respuesta a la antigua necesidad de reportes en el software de gestión empresarial por tratarse de una solución rápida y sencilla que ofrece gran atractivo visual. Los ejecutivos pueden explorar la información de manera rápida y fácil entre cantidades gigantescas de datos lo cual les permite tomar mejores decisiones, ahorrar tiempo en comparación a la ejecución de varios informes, la identificación rápida de los valores extremos, la creación de informaciones visuales, gráficas en pequeños espacios de tiempo y nuevas series analíticas como presupuestos, pronósticos y otras. (4)

Por otra parte, la web en tiempo real es un paradigma basado en dar información a los usuarios tan pronto como esté disponible, en lugar de exigir que ellos o sus programas verifiquen la fuente de los datos periódicamente en busca de actualizaciones. Se puede activar de muchas maneras diferentes y pueden requerir de una arquitectura técnica variable. Los sistemas de tiempo real están siendo implementados en las redes sociales, búsqueda, noticias y otros sitios, haciendo esas experiencias más parecidas la mensajería instantánea y facilitando innovaciones impredecibles. Los primeros beneficios incluyen una mayor participación del usuario y la disminución de las cargas del servidor; la web en tiempo real puede llegar a convertirse en omnipresente, un requisito para casi cualquier sitio web o servicio. (5)

Cuba, dada la necesidad de una independencia tecnológica y la importancia que tiene el desarrollo de sistemas informáticos ha empezado a desarrollar sus propios sistemas web. Con el objetivo de cumplir

esta tarea surge la Universidad de las Ciencias Informáticas (UCI), que desde sus inicios desarrolla productos con el fin de informatizar los sectores de la sociedad.

La UCI cuenta con un Centro de Informatización de la Gestión de Entidades (CEIGE), donde se desarrollan productos para la gestión empresarial. El análisis de información que se realiza se exporta al usuario en forma de reportes electrónicos en el Formato de Documentos Portables (PDF[3]), muy pobres en información visual, poco personalizables y referidos a un estado particular del sistema.

El marco de trabajo Sauxe es la plataforma base que utiliza el CEIGE para implementar aplicaciones web de gestión. Sauxe es un marco de trabajo orientado a componentes, que permite una mayor estandarización, integración, flexibilidad y agilidad en el proceso de desarrollo de aplicaciones web de gestión.

Actualmente, el marco de trabajo Sauxe no permite mostrar en tiempo real la información generada por los procesos de análisis de información de las aplicaciones desarrolladas sobre él, con lo que se dificulta la toma de decisiones de sistemas tan complejos dedicados a la planificación de recursos empresariales y estadísticos, por citar algunos de los ya construidos sobre este.

Según lo descrito anteriormente se define el **problema de la investigación** de la siguiente manera: ¿Cómo mostrar gráficamente y en tiempo real la información generada por las aplicaciones del CEIGE desarrolladas con el marco de trabajo Sauxe?

El **objeto de estudio** en el cual se enmarca el problema anteriormente planteado es: Dashboards para el monitoreo en tiempo real.

Se plantea como **objetivo general del trabajo**: Desarrollar una aplicación informática que permita la administración, configuración y visualización en tiempo real de la información generada por aplicaciones del CEIGE desarrolladas con el marco de trabajo Sauxe.

Para dar cumplimiento al objetivo general se plantean los siguientes **objetivos específicos**:

- Realizar el marco teórico de la investigación para definir las herramientas y métodos a utilizar en la construcción de la solución.
- Realizar la ingeniería de requisitos de la aplicación.

[3] Portable Document Format

- Desarrollar los componentes propuestos para el correcto funcionamiento del sistema.
- Validar la solución para garantizar que cumpla con los requisitos propuestos.

El **campo de acción** es la administración, configuración y visualización de información en plataformas de monitoreo en tiempo real.

La presente investigación tiene como **idea a defender:** El desarrollo de una aplicación informática que permita la administración, configuración y visualización en tiempo real para aplicaciones desarrolladas con el marco de trabajo Sauxe permitirá mostrar gráficamente y en tiempo inmediato la información generada por las aplicaciones del CEIGE desarrolladas bajo este marco de trabajo.

Los métodos de investigación utilizados fueron:

Métodos Teóricos

- **Histórico lógico:** Se utiliza para realizar una revisión histórica del desarrollo de los dashboard y constatar teóricamente cómo ha evolucionado la administración y configuración de la información que estos muestran.
- **Analítico-sintético:** Permitió el procesamiento de la información y arribar a las conclusiones prácticas y teóricas de la investigación.

Métodos Empíricos

- **Experimento:** Favoreciendo el desarrollo de pruebas para la verificación de las funcionalidades implementadas, con el fin de detectar errores y comprobar su correcto funcionamiento.
- **Observación:** Se utiliza para realizar una evaluación de la situación problemática en cuestión.

Este documento se encuentra compuesto por tres capítulos distribuido de la siguiente manera:

- Capítulo 1: Fundamentación Teórica de la investigación. En este capítulo se brinda una descripción general del objeto de estudio y los sistemas existentes asociados al campo de acción. Se plantea un análisis varios conceptos asociados al objeto de estudio, se definen las técnicas, tecnologías y metodologías que se utilizan para dar solución al problema planteado.

- Capítulo 2: Propuesta de Solución**:** Se exponen los procesos de negocios y requisitos a cumplir por la herramienta incluyendo los artefactos generados durante el diseño e implementación de la misma.

- Capítulo 3: Validación de la Solución Propuesta: Se exponen las pruebas utilizadas para la validación de la solución propuesta así como los resultados arrojados.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. INTRODUCCIÓN.

En el presente capítulo se definen una serie de conceptos necesarios para entender el objetivo fundamental del trabajo, se detallan las tendencias y tecnologías actuales, metodologías y herramientas para el desarrollo de la solución propuesta. Se profundiza las características de los dashboard actuales y los sistemas de tiempo real.

1.2. MARCO CONCEPTUAL.

1.2.1. DASHBOARD

En español no existe una traducción única al término **dashboard,** al traducir dashboard al idioma español obtenemos una variedad de posibles sinónimos, entre los más comunes encontramos: cuadro de mando, panel o tablero de instrumentos, panel o tablero de control. Para evitar ambigüedades se opta por utilizar el término en inglés, el cual es utilizado en la mayoría de la bibliografía consultada.

Los dashboard están diseñados para ayudar a interpretar el estado instantáneo de un elemento de control. Para proveer de un medio para controlar con rapidez el estado de cualquier sistema, los dashboard deben ser diseñados de manera particular para aprovechar los puntos fuertes de la percepción visual y el conocimiento. (6)

Según Stephen Few, presidente de Perceptual Edge y experto en visualización, un dashboard se puede definir como "*Una presentación visual de información importante, necesaria para lograr uno o más objetivos; consolidada y arreglada en una sola pantalla, de tal manera que la información pueda monitorearse con un vistazo."* (3)

En los sistemas de gestión de la información, un dashboard es un sistema de información ejecutivo, con una interfaz de usuario diseñada para ser fácil de interpretar.

1.2.2. SISTEMAS DE TIEMPO REAL.

Existen muchas interpretaciones de la naturaleza exacta de un Sistema en Tiempo Real (STR), sin embargo, todos tienen en común la noción de tiempo de respuesta.

El Diccionario Oxford de Computación ofrece la siguiente definición de sistema en tiempo real:

Un sistema en tiempo real es cualquier sistema donde el tiempo en que se produce su salida no es significativo. Por lo general debido a que la entrada corresponde a algún instante del mundo físico y la salida tiene relación con ese mismo instante. El retraso transcurrido entre la entrada y la salida debe ser lo suficientemente pequeño para considerarse una respuesta puntual. (7)

Otra definición de sistema de tiempo real dada por Stephen J. Young:

Cualquier actividad o sistema de procesamiento de información, que responda a un estímulo externo dentro de un período determinado y finito. (8)

El proyecto PDCS [4] el cual tiene como objetivo contribuir al proceso de diseño y construcción de sistemas confiables de computación mucho más previsibles y rentables, brinda la siguiente definición: (9)

Un sistema de tiempo real es un sistema que se requiere para reaccionar a los estímulos del entorno, dentro de los intervalos de tiempo determinado por el entorno.

Las acciones de los Sistemas de Tiempo Real se producen dentro de unos intervalos de tiempo determinados por la dinámica del sistema físico que supervisan o controlan. En su sentido más general, todas estas definiciones abarcan una gama muy amplia de actividades informáticas.

Los sistemas de tiempo real pueden ser divididos en dos grupos: (9)

Hard (Duro): Son aquellos en los que es absolutamente imperativo, que las respuestas se produzcan dentro del plazo especificado. Una respuesta tardía puede tener consecuencias fatales.

Soft (Suave): Son aquellos con restricciones de tiempo en las que una respuesta tardía no produce graves daños, el sistema seguirá funcionando correctamente.

En el presente trabajo de diploma cuando se usa el término “sistema de tiempo real” se refiere al sistema de tiempo real suave. Ya que es el término que se identifica con las características del dashboard a desarrollar.

En algunas ocasiones podemos ver referencias sobre sistemas de tiempo real cuando sólo se quiere decir que el sistema es rápido. Cabe mencionar que **tiempo real** no es sinónimo de rapidez, esto significa que no es el tiempo de respuesta lo que nos enfoca en un sistema de tiempo real, sino

[4] Predictably Dependable Computer Systems.

asegurarse que el tiempo de respuesta esté dentro del plazo especificado para resolver el problema al que el sistema está dedicado.

1.2.3. MARCO DE TRABAJO

Un marco de trabajo se puede definir como un conjunto de componentes que facilita y agiliza el desarrollo de sistemas. Con el término marco de trabajo se refiriere a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Esta una aplicación genérica incompleta y configurable a la que se le añade las últimas piezas para construir una aplicación concreta. (10)

Los objetivos principales que persigue un marco de trabajo son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. Entre algunos marcos de trabajo existentes podemos citar a Symfony patrocinado por Sensio Labs, Zend Framework desarrollado por Zend Technologies y Spring Framework desarrollado por Spring Source.

1.3. SISTEMAS ESTUDIADOS

Actualmente son muchas las empresas que para lograr una mejor gestión hacen uso de los dashboard, estos son diseñados con disímiles herramientas. Algunas de estas herramientas se seleccionan como objeto de estudio para analizar el tipo de información que manejan, además de las funcionalidades que presentan.

1.3.1. PENTAHO DASHBOARD

Conceptualmente, Pentaho Dashboard es una plataforma integrada para proporcionar información sobre datos, donde se puede ver todo tipo de informes, gráficos interactivos y los cubos creados con las herramientas de Pentaho como Pentaho Report Designer. Se trata de una interfaz que ofrece una vista centralizada sobre los movimientos de datos profesionales, lo que permite seguirlos y tomar decisiones. (11)

Sistemas Operativos compatibles: Windows, Macintosh, Unix y Linux.

Características

- Pentaho Dashboards está licenciado bajo GNU General Public License Version 2 (GPLv2).

- Incluye componentes de navegación y el visor de informes y análisis, que pueden ser integrados en los portales o páginas web.
- Genera contenido XML.
- Totalmente personalizable mediante definiciones XML de diseño, y hojas de estilo XSL y CSS.
- Identificación de métricas clave (KPIs, Key Performance Indicators), mediante la generación de Monitoreo/Métricas.
- Realización de investigaciones de detalles subyacentes, con reportes de soportes.
- Ejecución de seguimientos de excepciones, permitiendo pre-establecer alertas basadas en reglas del negocio.
- Gran variedad gráficos, tablas y velocímetros.
- Análisis OLAP.
- Capacidad de exportar a HTML, PDF, XLS, DOC, TIFF, CSV, XML.

1.3.2. OPEN REPORTS

Open Reports es una poderosa y flexible solución de reportes web de código abierto bajo licencia GPL. Ofrece generación de informes dinámicos y capacidades de programación de informes. Permite la creación de gráficas a través de ChartReports.

ChartReports utiliza JFreeCharts, un paquete abierto de gráficos, para proporcionar la capacidad de generar de forma dinámica gráficos sin la necesidad de escribir ningún código. Con el fin de crear un ChartReport debe crear una definición de tabla y luego agregar la tabla y los parámetros de consulta de gráficos para el informe. (12)

Características:

- Crear gráficos a partir de varios informes.
- Permite consultas SQL para obtener los datos a mostrar.
- Soporta 5 tipos de gráficos, Barras, Pastel, Anillo, Tiempo y Línea.

- Paneles colapsables de Alerta y Reportes.
- Soporte para una amplia variedad de formatos de exportación incluyendo PDF, HTML, CSV, XLS, RTF e imagen.

1.3.3. IBM COGNOS 8 GO! DASHBOARD.

El IBM Cognos 8 es una plataforma que sustenta una amplia gama de capacidades de gestión de rendimiento, tales como informes, dashboard y planificación para proporcionar información completa y oportuna a su comunidad de usuarios. (13)

IBM Cognos 8 Go! Dashboard integrado en la plataforma Cognos 8, traduce la información y datos en forma visual, presentaciones sofisticadas utilizando indicadores, mapas, gráficos y otros elementos para mostrar los resultados. Permite crear e interaccionar con dashboard complejos y confiables a través de una interfaz dinámica basada en flash. Se extiende más allá de solo medidores y gráficos al integrar IBM Cognos-Built, elementos externos como canales RSS[5], páginas web y facilidades de búsqueda. Disponible para Linux y Windows. (14) (15)

Características.

IBM Cognos 8 Go! Dashboard funciona en dos modos básicos Ensamblaje e Interactivo:

Ensamblaje.

Los usuarios de negocio o de tecnología de información pueden construir dashboard de partes pre-existentes.

- Crear y editar gráficos dinámicos y paneles de instrumentos que son fáciles de configurar.
- Adicionar reportes, informes parciales o indicadores de medidas arrastrándolos desde el panel de contenido.
- Agregar filtros a partir de menús pre-definidos a los elementos del dashboard (reportes, etc.) para mostrar exactamente la información necesaria.

Interactivo.

[5] *Really Simple Syndication*

- Cualquier usuario a cualquier nivel puede visualizar e interactuar con el dashboard.
- Modificar el diseño gráfico de un reporte o informe.
- Ordenar y filtrar los datos del dashboard para ajustarlos a los requerimientos necesarios.

Datos y gestión del ciclo de vida: El modo Interactivo y Ensamblado permite añadir extensiones IBM Cognos y organizar las conexiones a los datos en un solo lugar para la fácil administración.

Brinda conexión en directo con fuentes confiables de datos corporativos sujetos a la gestión habilitada por la plataforma de seguridad. Así como su integración con servicios web.

Beneficios.

- Funciones de gráficos enriquecidos para crear ágilmente paneles de instrumentos visualmente atractivos.
- Permite que los usuarios de la empresa ensamblen ellos mismos y personalicen los paneles de instrumentos.
- Se ajusta a la infraestructura existente

1.3.4. SAS BI DASHBOARD.

SAS[6] BI Dashboard permite a los usuarios monitorizar los indicadores clave de rendimiento. Los dashboard pueden incluir gráficos, texto, colores e hipervínculos. Son creados, modificados y visualizados a través de una fácil interfaz basada en web. Todo el contenido se muestra en un entorno basado en roles, es seguro, personalizable y extensible. (16)

Características.

SAS BI Dashboard Designer.

- Una nueva interfaz interactiva e intuitiva. Permite arrastrar y soltar objetos sobre el dashboard que se está diseñando.

[6] Statistical Analysis System.

- Puede cambiar el tamaño de los indicadores con el ratón. Las nuevas características ayudan a trazar el contenido del panel con precisión.
- El diseñador trabaja en tiempo real, mientras que se está diseñando, lo que se ve, es el producto final.
- El Dashboard Designer también ofrece ahora muchos más tipos de indicadores para su uso en un tablero de instrumentos, así como nuevos estilos de gráficos para mejorar el aspecto de los indicadores.

SAS BI Dashboard Viewer.

- Mediante la creación de dashboard, los diseñadores de proporcionan a los usuarios datos de su negocio. Los usuarios empresariales pueden ver estos dashboard en el SAS BI Dashboard Viewer.
- Compartir comentarios con otros usuarios acerca de los indicadores del dashboard, crear alertas personalizadas, y visualización de datos de una manera interactiva.
- Moverse fácilmente entre el Dashboard Designer y el Dashboard Viewer. Permite a los diseñadores crear un nuevo tablero de instrumentos, y sin esfuerzo verlo en el Dashboard Viewer, y volver rápidamente a la vez al Dashboard Designer. O bien, se puede utilizar la característica de vista previa y ver dashboard y la mayoría de sus características sin salir del Dashboard Designer en absoluto.
- Exportar a los formatos, Excel, PDF de Adobe.

Beneficios.

- Los paneles son fáciles de diseñar y construir.
- Arrastrar y soltar hace que el proceso sea rápido y sencillo.
- Los indicadores son fáciles de adaptar y cambiar de tamaño.

- Utilizan una amplia variedad de fuentes de datos: Mapas de información, Procesos almacenados, Consultas SQL[7], Tablas.
- Fácil colaboración e intercambio de información. Permite comentar los indicadores logrando una comunicación rápida.
- La función de alerta hace que sea fácil para los usuarios de negocio para vigilar las áreas clave y de ser notificado de forma proactiva de las condiciones que requieren seguimiento.
- Indicadores monitorizados frecuentemente son fáciles de recuperar gracias a la opción Favoritos.

1.3.5. WEBFOCUS.

Information Builders ofrece dashboard robustos e innovadores que permiten compartir las misiones con toda la empresa, enlazar las operaciones directamente a los objetivos estratégicos, y monitorizar en tiempo real.

WebFOCUS dashboard, proporciona información sobre la organización y su funcionamiento, mejorando la toma de decisiones de los ejecutivos, y trabajadores de primera línea. WebFOCUS dashboard se utiliza para descubrir oportunidades, identificar las tendencias y encontrar vulnerabilidades antes de que se conviertan en problemas. (17)

Características.

Permite a los usuarios finales en todos los niveles crear sus propios dashboard usando las tecnologías más innovadoras y modernas.

Las capacidades de personalización de WebFOCUS permiten:

- Ensamblaje de arrastrar y soltar.
- Soporte para Herramientas Web 2.0 de terceros.
- Difusión y sincronización de las preferencias hacia otros objetos del dashboard.
- Almacenamiento inteligente en cache de los datos del dashboard.

[7] Structured Query Language (Lenguaje de Consulta Estructurado).

- Exportar en varios formatos, Excel, PDF de Adobe, Flash, Adobe Flex PDF y Microsoft PowerPoint.
- Calendario para entrega vía correo electrónico, un archivo de reporte, impresora y sitio FTP[8].
- Maximizar la comunicación de información a través del más completo conjunto de datos y visualizaciones de información en la industria, Más de 200 tipos de gráficos como Pareto, diagramas de dispersión, diagramas de constelación, las nubes de etiquetas, entre otros.
- Incluye mejoras de extracción, transformación y carga en tiempo real de datos desde bases de datos, y mapas de información, así como integración total con adaptadores de datos en tiempo real.

1.3.6. STYLE SCOPE FREE EDITION

Style Scope Free Edition es un servidor libre de pequeñas dimensiones desarrollado en Java, que ofrece dashboards interactivos basados en la Web, con visualizaciones en flash. Disponible para Windows, Unix o Mac con la posibilidad de conectarse a los datos de bases de datos relacionales, así como hojas de cálculo. Los tipos de dashboards que se pueden crear van desde los utilizados para monitorizar hasta los más sofisticados dashboards de análisis y de gestión empresarial. El dashboard no puede ser utilizado para propósitos comerciales. (18)

Características.

- Los paneles son compartibles dentro de una LAN o WAN.
- Arrastre y suelte las hojas de cálculo como diseñador.
- Gráficos multidimensionales, tales como gráficos de burbujas.
- Exploración de datos (destacando los datos relacionados a través de varios gráficos)
- Cartografía geográfica avanzada.
- Datos en tiempo real.
- Calendarios, deslizadores del área de distribución, indicadores, referencias cruzadas.

[8] File Transfer Protocol - Protocolo de Transferencia de Archivos

- Exportación de gráficos, con alta fidelidad a Excel, PowerPoint y PDF.

Requisitos Técnicos.

- Cualquier sistema operativo con Java Virtual Machine 1.6 o superior.
- 200 MB de espacio en disco.
- 1 GB de RAM.
- Cualquier base de datos relacional con un driver JDBC, como Microsoft Access, MySQL, Oracle.

A continuación se muestra un resumen de algunas características de los sistemas estudiados.

Tabla I: Resumen Características – Sistemas

	SO	Licencia	Portable	Formatos	Fuentes de datos	Diseño en tiempo real
PENTAHO Dashboard	Windows, Macintosh, Unix y Linux.	GPLv2	Parte de un sistema	HTML, PDF, XLS, DOC, TIFF, CSV, XML	BD, Cubo OLAP, Pentaho metadata, XML, BD	No
OpenReports	Windows	GPL	Independiente	PDF, HTML, CSV, XLS, RTF,imagen	Consultas SQL, BD	No
IBM GO	Windows, Linux	Propietaria	Parte de un sistema.	PDF, Excel, Word, Power Point	BD. Servicios	No
SAS BI	Windows, Linux	Propietaria	Independiente.	Excel, PDF de Adobe.	Mapas de información, BD	No

WebFocus	Windows	Propietaria	Independiente	Excel, PDF, Flash, PowerPoint.	BD, mapas de información	Sí
Style Scope	Windows, Linux, Mac	Libre	Independiente	Microsoft Office, Flash, PDF, Adobe AIR	Cualquier base de datos relacional, culo Excel	Sí

Entre las principales limitaciones presentadas por estos sistemas se encuentran:

- Formar parte de otro sistema o plataforma.
- No son soluciones de código abierto.
- No han sido desarrollados utilizando PHP, ExtJS y PostgreSQL 3.8. Tecnologías necesarias para poder incluirlo como una herramienta del marco de trabajo Sauxe.

1.4. MENSAJERÍA INSTANTÁNEA

En los últimos años el acceso a la información en tiempo real se ha vuelto de vital importancia, tanto en el ámbito laboral como en el personal. Las personas necesitan que la distribución y/o el intercambio de la información sea inmediata y segura. Una de las herramientas más utilizadas para llevar a cabo esta tarea es la mensajería instantánea, por ser la que más ventajas ofrece. Para realizar el intercambio de información a través de la mensajería instantánea, es necesario establecer un conjunto de reglas a seguir para que los equipos involucrados se comuniquen haciendo uso del mismo idioma. A esto se le denomina protocolo. A continuación veremos no de los protocolos más utilizados, el XMPP. (19)

XMPP

El protocolo XMPP[9] usa una arquitectura cliente-servidor descentralizada, lo que permite a los desarrolladores de programas clientes enfocarse en la experiencia de los usuarios, y a los desarrolladores de servidores enfocarse en el rendimiento y escalabilidad. XMPP es en esencia una tecnología para transmitir XML de una entidad a otra cercano al tiempo real. (20)

[9] eXtensible Messaging and Presence Protocol

XMPP proporciona los siguientes servicios:

- Codificación de canal: Este servicio brinda encriptación de las conexiones entre un cliente y un servidor, o dos servidores, para mayor seguridad de las aplicaciones.

- Autenticación: Este servicio es definido para desarrollo de aplicaciones seguras. En este caso, el servicio de autenticación asegura que los entes intentando comunicarse deben ser primero autenticados por un servidor que actúa como cierto portero para acceso de red.

- Presencia: La función de este servicio es descubrir sobre la disponibilidad de red de otros entes. Un servicio de presencia responde la pregunta, ¿Está la entidad en línea, disponible para la comunicación offline o no disponible? Los datos de presencia pueden también incluir información más detallada tal como si una persona se encontrara en una reunión. Típicamente, el compartir información de presencia está basado en un sistema de subscripción entre dos entes a fin de proteger la privacidad del usuario.

- Mensajería uno a uno: El propósito de este servicio es enviar mensajes a otra entidad. El uso clásico de uno a uno enviando mensajes que pueden ser arbitrarios XML. Cualesquiera dos entes en una red pueden intercambiar mensajes, ellos pueden ser servidores, componentes, dispositivos o servicios de red de XMPP habilitados.

- Entorno Peer-to-Peer para sesiones de medios: Este servicio permite negociar y manejar una sesión de medios con otra entidad. La sesión puede ser usada para el propósito de charla de voz, charla de video, transferencia de archivo y otras interacciones en tiempo real.

Una de las principales ventajas de este protocolo es que, a diferencia de otros protocolos de mensajería, se trata de un estándar libre.

BOSH

En esencia, BOSH[10] proporciona una capa de emulación de datos bidireccional, sincrónica. Permite establecer una conexión a un servidor XMPP desde HTTP. Esto es posible porque XMPP establece una conexión persistente. Se mantiene la conexión desde el principio hasta el final, y mientras se envían todos los mensajes que quieran. BOSH es un estándar sumamente nuevo y se están creando

[10] Bidirectional streams Over Synchronous HTTP.

librerías para su optimización. Actualmente brinda grandes facilidades para proyectos de tiempo real y es por ello que se decide poner en práctica para esta arquitectura.

- Puede transportar pequeñas instancias de XMPP sobre HTTP.
- Aumenta la escalabilidad de las aplicaciones usando XMPP.
- Los servidores de XMPP pueden manipular TONELADAS de usuarios concurrentes.
- Puede ser usado en situaciones dónde el servidor XMPP no está disponible.
- No pierde datos. Los mensajes se guardarán en el servidor hasta que el cliente esté nuevamente.
- BOSH define qué tan arbitrario pueden ser los elementos XML transportados eficazmente y de fuente fidedigna sobre el HTTP en ambas direcciones entre un cliente y servidor. (21)

STROPHEJS

Strophejs es una librería JavaScript que facilita conexiones TCP persistentes, esta librería se basa en BOSH para emular la persistencia, con estado de conexión, en ambos sentidos a un servidor XMPP.

Strophejs es una API pequeña y fácil de entender, ya desde hace algunos años ha sido probado en los principales navegadores y su comunidad de desarrolladores ha ido aumentando para beneficio de los que se inician en su utilización. Esta librería detecta muchos errores e incluye optimizaciones que ningún otro marco de AJAX permite hacer, ofreciendo así excelentes rendimientos. Es una excelente librería para implementar clientes XMPP. (22)

1.5. TECNOLOGÍAS UTILIZADAS

Las tecnologías son un conjunto de conocimientos técnicos que son utilizados para el desarrollo de software, que posibilitan la satisfacción y adaptación de las necesidades a las que se enfrentan los desarrolladores. El presente trabajo forma parte del proceso productivo del CEIGE, el cual ha tomado decisiones tecnológicas que involucran la utilización de tecnologías de código abierto para el desarrollo de sus productos. A continuación se describen brevemente estas tecnologías.

1.5.1. MARCOS DE TRABAJO Y LIBRERÍAS.

Sauxe

El desarrollo de la solución se realiza utilizando el marco de trabajo Sauxe desarrollado por el CEIGE, el cual contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo (23). Es un producto desarrollado sobre tecnologías libres como el lenguaje PHP, gestor de base de datos Postgres, servidor web Apache, etc. Su administración centralizada de todos los aspectos necesarios a tener en cuenta en el desarrollo de aplicaciones de gestión lo convierte en un producto de punta en esta rama. Permite realizar un número de funcionalidades que hace esta arquitectura muy aplicable para cualquier entorno web en PHP. Permite la gestión de multi-entidad y con esta la compartimentación de la información de cada una de ellas.

Sauxe está compuesto por varios marcos de trabajo, los cuales serán descritos a continuación, estructurados en niveles o capas como se puede apreciar en la siguiente figura.

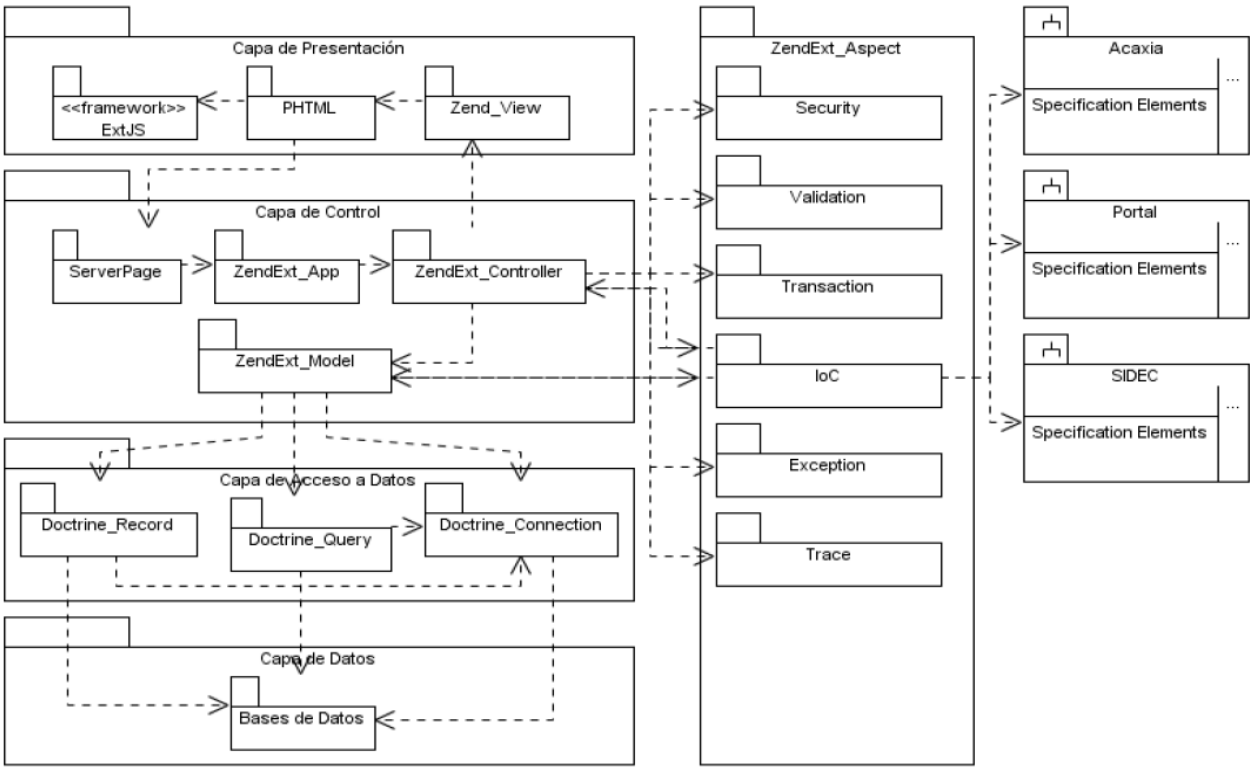


Figura 1: Estructura del Marco de Trabajo Sauxe

Zend Framework

Zend Framework es un marco de trabajo de código abierto para el desarrollo de aplicaciones y servicios web con PHP 5. Zend Framework se implementa utilizando 100% código orientado a objetos. La estructura de los componentes de Zend Framework es algo único, cada componente ha sido diseñado con pocas dependencias de otros componentes lo que permite usar los componentes de forma individual. Ofrece una solución de alto rendimiento y una robusta implementación MVC, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores pueden consolidar todas estas operaciones en una interfaz orientada a objetos sencilla de usar (24).

Ext JS

Ext JS es una librería JavaScript ligera y de alto rendimiento, que nos permite crear páginas e interfaces web dinámicas. Provee interfaces gráficas de usuario que brindan experiencias parecidas o iguales a las que se tienen con aplicaciones de escritorio. Es extensible para la gran mayoría de los navegadores, evitando el tedioso problema de validar el código para cada uno de estos. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, la carga de procesamiento se distribuye permitiendo que el servidor, al tener menor carga, pueda manejar los clientes de manera más eficiente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de estar sujeta a una acción al usuario, dando la libertad de cargar la información sin que este lo note (25).

Doctrine

Doctrine es un potente y completo sistema ORM[11] para PHP con un DBAL[12] incorporado el cual permite trabajar con un esquema de base de datos como si fuese un conjunto de objetos. Está inspirado en Hibernate, que es uno de los ORM más populares y grandes que existen y nos brinda una capa de abstracción de la base de datos muy completa. Posee la habilidad de escribir opcionalmente las preguntas de la base de datos utilizando la programación orientada a objetos debido a que doctrine utiliza el patrón Active Record para manejar la base de datos. Esto les brinda una alternativa poderosa a diseñadores de SQL que mantiene un máximo de flexibilidad sin requerir la duplicación del código innecesario. La característica más importante es que te da la posibilidad de escribir consultas de base de datos en un lenguaje propio llamado Doctrine Query Language (DQL) (26).

[11] Object-Relational Mapping. (Mapeado Relacional de Objetos).
[12] Siglas en ingles de Capa de Abstracción de Bases de Datos

1.5.2. LENGUAJES DE PROGRAMACIÓN.

PHP

PHP (acrónimo de Hypertext Preprocessor) es un lenguaje interpretado de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Sus sintaxis son muy similares a lenguajes como C y PERL. Es multiplataforma permitiendo migrar las aplicaciones de un sistema a otro sin necesidad de realizar cambios en el código. Su rapidez en la ejecución y los bajos requerimientos de consumo en los sistemas donde es desplegado lo hacen uno de los preferidos por los desarrolladores.

Dispone de una conexión nativa a los principales sistemas de base de datos utilizados actualmente tales como Postgres, MySQL, Oracle, Microsoft SQL Server, lo cual permite la creación de aplicaciones web robustas. Su mayor ventaja radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso, además de contar con una comunidad de desarrolladores que intercambian experiencias lo que facilita la rápida solución de problemas sin costo alguno. Quizás una de sus mayores desventajas radica en que promueve la creación de código desordenado, por lo que lo hace muy complejo de mantener. (27)

JavaScript

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Este es un lenguaje Case Sensitive[13] y al contrario que Java (con el cual no existe ninguna relación), no es exactamente un lenguaje orientado a objetos, puesto que no dispone de herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base que serían los prototipos y extendiendo su funcionalidad. Su función es ampliar las funcionalidades de HTML, permitiendo interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Mozilla Firefox, Netscape, Opera (28).

XML

XML significa Extensible Markup Language[14], se utiliza para describir documentos y datos en un formato estándar, basado en texto que puede ser fácilmente transportado a través de protocolos

[13] Sensible a la diferencia entre minúsculas y mayúsculas.
[14] Lenguaje de Marcado Extensible, XML siglas en inglés.

estándar de Internet. Tanto XML, como HTML, se basan en la madre de todas los lenguajes de marcas, Standard Generalized Markup Language (SGML pos sus siglas en ingles).

XML es un lenguaje de marcas que ofrece un formato para la descripción de datos estructurados, el cual conserva todas las propiedades importantes del antes mencionado SGML. Es decir, XML es un metalenguaje, dado que con él podemos definir nuestro propio lenguaje de presentación y, a diferencia del HTML, que se centra en la representación de la información, XML se centra en la información en sí misma. La particularidad más importante del XML es que no posee etiquetas prefijadas con anterioridad, ya que es el propio diseñador el que las crea a su antojo, dependiendo del contenido del documento (29).

Otros lenguajes y formatos utilizados

CSS

CSS[15] es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc. (30)

JSON

JSON [16] es un formato ligero de intercambio de datos, un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. Es un formato de texto que es completamente

[15] Hojas de estilo cascada, CSS siglas en inglés.

independiente del lenguaje, estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. (31)

UML 2.0

El proceso unificado de desarrollo (UML) representa un número de modelos de desarrollo basados en componentes que han sido propuestos en la industria. El proceso unificado define los componentes que se utilizarán para construir el sistema y las interfaces que conectarán los componentes. Utilizando una combinación del desarrollo incremental e iterativo, el proceso unificado define la función del sistema aplicando un enfoque basado en escenarios (desde el punto de vista del usuario) y acopla la función con un marco de trabajo arquitectónico que identifica la forma que tomará el software. (32)

1.5.3. HERRAMIENTAS

A continuación se describen las herramientas utilizadas para lograr el cumplimiento del objetivo general.

Visual Paradigm

Visual Paradigm (VP) es una herramienta de diseño UML y herramienta CASE[17] diseñada para ayudar al desarrollo de software. Utiliza UML como lenguaje de modelado y soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue. Ofrece un completo conjunto de herramientas para equipos de desarrollo de software, necesarias para la captura de requisitos, planificación de software, el modelado de clases, modelado de datos, etc.

Permite el modelado colaborativo con Subversion y la integración de aplicaciones empresariales a las bases de datos. Es capaz de realizar ingeniería tanto directa como inversa, posibilita generar código para lenguajes como PHP. Posee un generador de informes automático para la documentación del proyecto en varios formatos como HTML, PDF. La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto.

NetBeans IDE

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000

[16] JavaScript Object Notation
[17] Ingeniería de Software Asistida por Computadora, CASE siglas en inglés.

y continúa siendo el patrocinador principal de los proyectos. Al día de hoy hay disponibles dos productos: el NetBeans IDE y NetBeans Platform.

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. (33)

Git

Para la gestión de los cambios en el código se hace indispensable contar con una herramienta que facilite el control de versiones.

Git es un sistema de control de versiones distribuido, libre y de código abierto, diseñado para manejar desde pequeños hasta grandes proyectos, con velocidad y eficiencia. Nace como una necesidad para el desarrollo del kernel de Linux, desarrollado inicialmente por el propio Linus Torvalds, que deseaba un sistema rápido, eficiente y distribuido. (34)

Características de GIT. (35)

- Fuerte apoyo al desarrollo no-lineal, rapidez en la gestión de ramificaciones y mezclado de diferentes versiones. Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.
- Gestión distribuida. Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramificaciones adicionales y pueden ser fusionados en la misma manera que se hace con la ramificación local.

1.5.4. TECNOLOGÍAS.

Apache

Apache es un servidor web de código libre para la transferencia de hipertextos (Hypertext Transfer Protocol, HTTP por sus siglas en inglés) para plataformas Unix, Windows, y Macintosh. Su implementación se realiza de forma colaborativa, con prestaciones y funcionalidades equivalentes a las de los servidores comerciales. El proyecto está dirigido y controlado por un grupo de voluntarios de

todo el mundo que, usando Internet y la web para comunicarse, planifican y desarrollan el servidor y la documentación relacionada (36).

Principales características:

- Es un servidor de web conforme al protocolo HTTP/1.1
- Soporta tanto host basados en IP como host virtuales.
- Apache soporta autentificación básica basada en la Web.
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.
- Personalización de las respuestas ante los posibles errores que se puedan dar en el servidor. (37)

Base de Datos PostgreSQL.

PostgreSQL es un potente Sistema de Base de Datos Objeto-Relacional (ORDBMS por sus siglas en inglés) libre, bajo licencia Berkeley Software Distribución (BSD). La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre. Funciona en todos los sistemas operativos importantes, incluyendo Linux, UNIX (AIX, HP-UX, SGI IRIX, Mac OS, Solaris, Tru64), y Windows. (38)

PostgreSQL cuenta con características sofisticadas como el Control de Concurrencia Multi-versión (MVCC por sus siglas en ingles), recuperación a punto, espacios de tablas, replicado asíncrono, transacciones anidadas (con puntos de guardado), copias de seguridad en línea o en caliente, un planificador/optimizador sofisticado de consultas y un sistema de registros con escritura adelantada para mayor tolerancia a fallos. Soporta conjuntos de caracteres internacionales, codificación de caracteres y está preparado para ordenado de acuerdo a la internacionalización, así como sensible a capitalización y formateo. Es altamente escalable en cuanto a cantidad de datos y usuarios. (39)

Propiedades:

- Atomicidad, asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- Consistencia, asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- Aislamiento, asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.

Durabilidad, asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

1.6. CONCLUSIONES

Con el estudio de algunos dashboard existentes en el mundo se llega a la conclusión de que los mismos han demostrado el gran avance tecnológico que existe en la actualidad en cuanto a este tema, capaces de facilitar el acceso, extracción y análisis rápido de los datos, aunque con la limitante de que por muy ventajoso y moderno que sea un producto no siempre se ajusta a los requerimientos de una institución determinada.

Los dashboards estudiados:

- No satisfacen las exigencias necesarias para su integración con el marco de trabajo SAUXE.
- No permiten la configuración de la fuente de datos para obtención de la información a través del protocolo XMPP.
- Sirven de guía de referencia a la hora de definir cuáles son las principales funcionalidades que debe cumplir el sistema a desarrollar.

Por tanto se decidió elaborar una aplicación para el marco de trabajo SAUXE, que permita a los usuarios visualizar gráficamente los datos generados por aplicaciones del CEIGE. Dicha herramienta será implementada siguiendo las decisiones tecnológicas tomadas por el CEIGE, las cuales involucra la utilización de tecnologías de código abierto, así como la adopción del protocolo XMPP para la comunicación en tiempo real, por su robustez y que a diferencia de otros protocolos de mensajería se trata de un estándar libre.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN

2.1. INTRODUCCIÓN

En el siguiente capítulo se muestra la descripción de la herramienta propuesta bajo el nombre WaoX Dashboard. Primeramente se describen cada uno de los procesos de negocio mediante los cuales se capturan los requisitos necesarios para la realización del sistema, se enumeran y especifican cada uno de ellos, tanto los funcionales como los no funcionales. Posteriormente se muestran los artefactos correspondientes al diseño de clases y modelo de datos. Por último se realiza la descripción del diseño de la solución que se elaboró para darle solución a los objetivos propuestos.

2.2. DESCRIPCIÓN DEL SISTEMA

WaoX Dashboard es una aplicación para monitorizar la información generada por las diferentes aplicaciones del CEIGE desarrolladas con el marco de trabajo Sauxe. WaoX fue diseñado con el principio de brindar a los miembros de la organización de una herramienta que facilite el seguimiento de la información generada en tiempo real, mejorando así la toma de decisiones.

Para complementar lo antes descrito el sistema permite personalizar el ambiente de trabajo, permitiendo adicionar o eliminar monitores, cada monitor puede mostrar varias vistas (gráficos y tablas) con información en tiempo real brindado por las fuentes de datos facilitadas por el usuario. Se pretende cubrir totalmente o en su parcialidad los problemas de análisis de información en tiempo real que presentan las aplicaciones desarrolladas sobre Sauxe.

El sistema brinda una interfaz compuesta por una barra de menú, donde el usuario puede seleccionar diferentes funcionalidades una vez creado un monitor, estos se irán ubicando en un panel lateral izquierdo, que cuenta en su parte inferior con una barra de herramientas con dos botones, que permiten adicionar y eliminar un monitor. Una vez adicionado un monitor se puede crear una vista, permitiendo hasta 4 vistas por monitor. Las fuentes de datos serán adicionadas por el usuario, mediante una interfaz donde le permita ingresar el código, nombre y una breve descripción. Al ser adicionada la fuente de datos permanecerá en espera de un mensaje con su código, si el formato es correcto pasará a estado de lista para su configuración al adicionar una vista. Cada vista estará asociada a una fuente de datos mediante el código.

Para crear una vista es necesario haber adicionado una fuente de datos, estas cuentan con un nombre, una fuente de datos y un tipo, este último define el modo de visualización de los datos (línea,

columna, pastel y tabla). Una vez elegido el tipo y la fuente de datos se podrá pasar a la configuración de la vista según los datos recibidos de la fuente de datos elegida.

Los mensajes recibidos de las fuentes de datos tienes que cumplir con la estructura definida en el siguiente epígrafe.

2.3. FUENTE DE DATOS

Las aplicaciones del marco de trabajo Sauxe para la comunicación con WaoX, deberán configurar las fuentes de datos cumpliendo con la estructura de datos que se define a continuación, logrando así, establecer un estándar que permita un óptimo mapeo entre las fuentes de datos y las vistas. Como lenguaje de intercambio de datos se decidió utilizar JSON, por poseer un formato ligero para el intercambio de datos, fácil de interpretar y generar.

Definición de la estructura de las fuentes de datos.

La estructura de la fuente de datos para el intercambio de información se define como sigue:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estructura: **{ code :** Código **, msg : [** Mensaje **] }**

Donde:

- Código: Será el identificador de la fuente de datos, por lo que no puede existir dos fuentes de datos en el marco de trabajo Sauxe con igual código. Se expresará mediante un valor alfanumérico. Ejemplo: 001, b12, F12ced.
- Mensaje: Contendrá los campos (fields) de la fuente de datos, la cantidad de tuplas (results), los datos de las tuplas (rows).

  { rows : [{ tupla1**} , {** tupla2 **} , {**tuplaN**}] , metaData: { root :** rows **, totalProperty:** results **, fields : [{ name :** nombre **} , { name :** nombre1 **} , { name :** nombreN **}] } }**

- Tupla**: { nombre :** valor1**, nombre1 :** valor1**, nombren :** valorN **}**

En la figura 2 se presenta un ejemplo:

```
{code: 123, msg: [
    {
        "rows": [
            {name:"Jul 2007", visits: 240, views: 3000000},
            {name:"Andrew", visits: 20000, views: 354000},
            {name:"Sep 07", visits: 000, views: 400},
            {name:"Oct 07", visits: 375000, views: 4000000},
            {name:"Nov 07", visits: 12, views: 400},
            {name:"Dec 0123", visits: 1000002, views: 580},
            {name:"Jan 08", visits: 520000, views: 6000},
            {name:"Feb 08", visits: 200, views: 14442}
        ],
        metaData: {
            "root": "rows",
            "totalProperty": 8,
            "fields": [
                {"name": "name"},
                {"name": "visits"},
                {"name": "views",}
            ],

        }
    }]}
```

Figura 2: Ejemplo fuente de datos WaoX.

Esta estructura provee facilidades de mapeo, brindando la posibilidad de poder configurar cualquier tipo de vista para que muestre los datos deseados de una determinada fuente de datos.

2.4. MODELO DE DOMINIO

El primer paso en el proceso de desarrollo de software es precisamente alcanzar cierto nivel de conocimientos sobre el problema en cuestión.

En el presente trabajo de diploma se intenta capturar los tipos más importantes de objetos que existen en el entorno del sistema, por lo que decide realizar un modelo de dominio. Para conformar dicho modelo se ha elaborado un modelo conceptual que contiene los objetos y conceptos que se enmarcan en la problemática expuesta, así como un diccionario de datos.

2.4.1. MODELO CONCEPTUAL

Para descomponer el dominio del problema hay que identificar los conceptos y las asociaciones del dominio que se juzgan importantes. El resultado puede expresarse en un modelo conceptual, que no

es más que una representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés. (40)

El modelo conceptual se representa usando el diagrama de objetos de UML como se muestra en la Figura 3. Modelo Conceptual. donde se observan conceptos del dominio y sus relaciones.

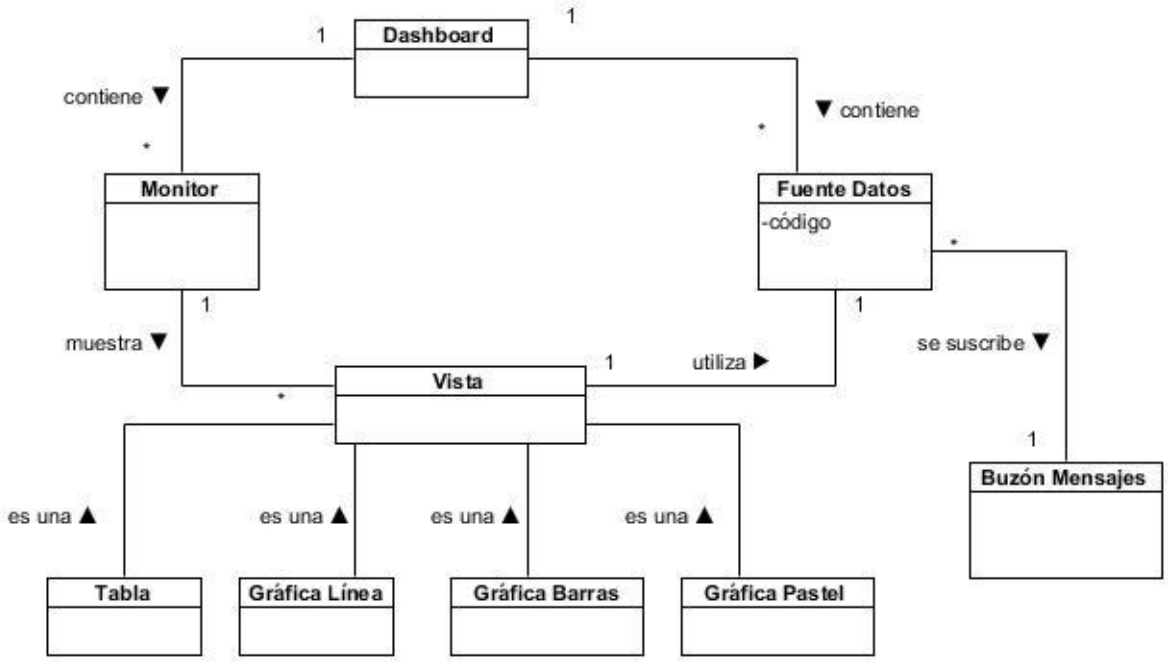


Figura 3. Modelo Conceptual.

2.5. REQUISITOS DEL SISTEMA

2.5.1. REQUISITOS FUNCIONALES

Los requisitos funcionales son características requeridas del sistema, que expresan una capacidad de acción del mismo o una funcionalidad; de manera que especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto. (41)

Constituyen un elemento clave para el diseño y la posterior implementación. Facilitan el entendimiento de los procesos a implementar permitiendo una buena comprensión del problema y posibilitando la identificación de las clases y las funcionalidades a desarrollar.

Tabla II: Requisitos Funcionales.

R.1 Gestionar fuentes de datos.	1.1 Adicionar fuente de datos.
	1.2 Eliminar fuente de datos.
R.2 Administrar los monitores.	2.1 Adicionar monitor.
	2.2 Eliminar monitor.
	2.3 Exportar tablas del monitor a hoja de cálculo.
R.3 Administrar las vistas.	3.1 Adicionar vistas.
	3.2 Eliminar vistas
	3.3 Definir el mapeo entre las fuentes de datos y las vistas.

Para lograr una mejor comprensión de la funcionalidad asociada a cada proceso es necesario realizar una descripción textual de los requisitos. A continuación se describirán los requisitos principales del sistema.

R1. Gestionar las fuentes de datos.

Especificación del requisito: Adicionar fuente de datos.

Precondiciones	NA
Flujo de eventos	
Flujo básico	
1	Se introducen de la fuente de datos los siguientes datos: Nombre Código Descripción
2	El sistema comprueba si no existen fuentes de datos con igual nombre o código.
3	Si los datos son correctos el sistema los registra.

4	Concluye el requisito.
Pos-condiciones	
1	Se registró en el sistema una nueva fuente de datos.
Flujos alternativos	
Flujo alternativo 2.a Existen fuentes de datos con el nombre o código insertado.	
1	El sistema notifica que existen fuentes de datos con el nombre o código insertado.
2	Volver al Paso 1.
Flujo alternativo 3.a Información incompleta.	
1	El sistema señala los datos vacíos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 3 del flujo básico.
Flujo alternativo *.a El usuario cancela la acción.	
1	Concluye el requisito.
Pos-condiciones	
1	NA
Validaciones	

Relaciones	**Requisitos Incluidos**	NA
	Extensiones	NA
Conceptos	**Fuente de Datos**	Visibles en la interfaz: Nombre Código Descripción
Requisitos especiales	NA	
Asuntos pendientes	NA	

Prototipo elemental de interfaz gráfica de usuario.

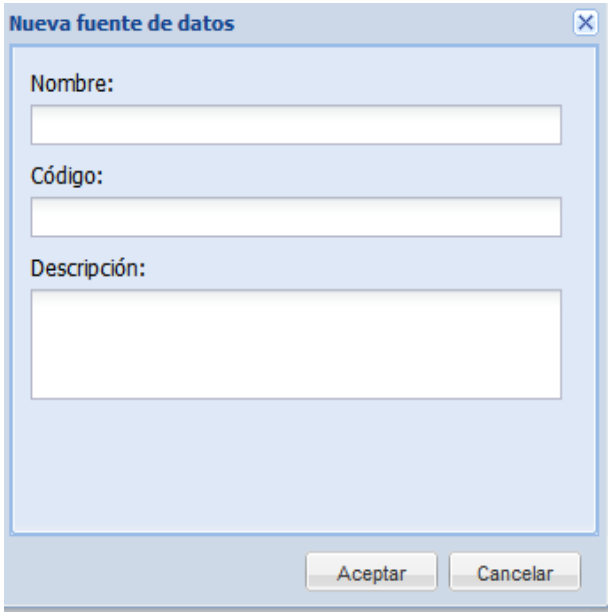


Figura 4: Interfaz Nueva Fuente de Datos.

Especificación del requisito: Eliminar fuente de datos.

Precondiciones	Debe existir al menos una fuente insertada
Flujo de eventos	
Flujo básico	
1	Se selecciona la fuente a eliminar.
2	Se solicita confirmación para eliminar la fuente de datos.
3	Concluye el requisito.
Pos-condiciones	
1	Se eliminó del sistema una fuente de datos.
Flujos alternativos	
Flujo alternativo 2.a El usuario no confirma la eliminación de la fuente de datos.	
1	El usuario cancela la eliminación.
2	Volver al paso 1.
Pos-condiciones	
1	NA
Validaciones	

	El estado de la fuente de datos no puede estar en uso.	
Relaciones	**Requisitos Incluidos**	NA
	Extensiones	NA
Conceptos	**Fuente de Datos**	Visibles en la interfaz: Nombre Código Descripción Estado
Requisitos especiales	NA	
Asuntos pendientes	NA	

Prototipo elemental de interfaz gráfica de usuario.

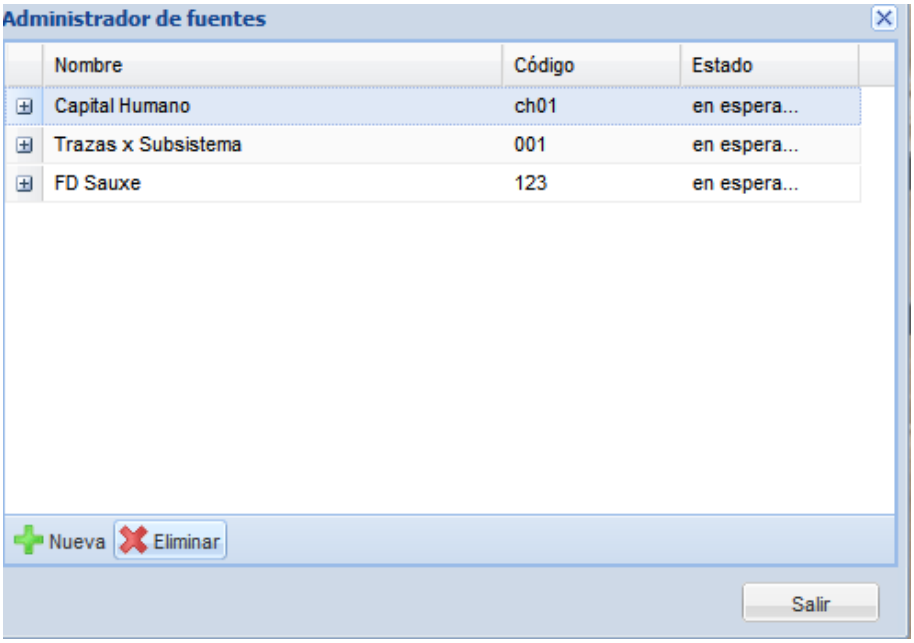


Figura 5: Interfaz Administrador de Fuentes.

R2. Administrar vistas.

Especificación del requisito: Adicionar vista.

Precondiciones		Debe existir al menos una fuente de datos configurable en el sistema.
Flujo de eventos		
Flujo básico		
1	Se insertan el nombre	
2	Se elige el tipo de vista.	
3	Se elige la fuente de datos.	
4	Se configura la vista.	
5	Si los datos son correctos el sistema registra la vista.	
6	Concluye el requisito.	
Pos-condiciones		
1	Se registró en el sistema una nueva vista.	
Flujos alternativos		
Flujo alternativo 5.a Información incompleta.		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 5 del flujo básico.	
Flujo alternativo *.a El usuario cancela la acción.		
2	Concluye el requisito.	
Pos-condiciones		
1	NA	
Validaciones		
	NA	
Relaciones	**Requisitos Incluidos**	Paso 4: Definir el mapeo entre la fuente de datos y la vista.
	Extensiones	NA

Conceptos	**Vista**	Visibles en la interfaz: Nombre Tipo Fuente de datos Propiedades.
Requisitos especiales	NA	
Asuntos pendientes	NA	

Prototipo elemental de interfaz gráfica de usuario.

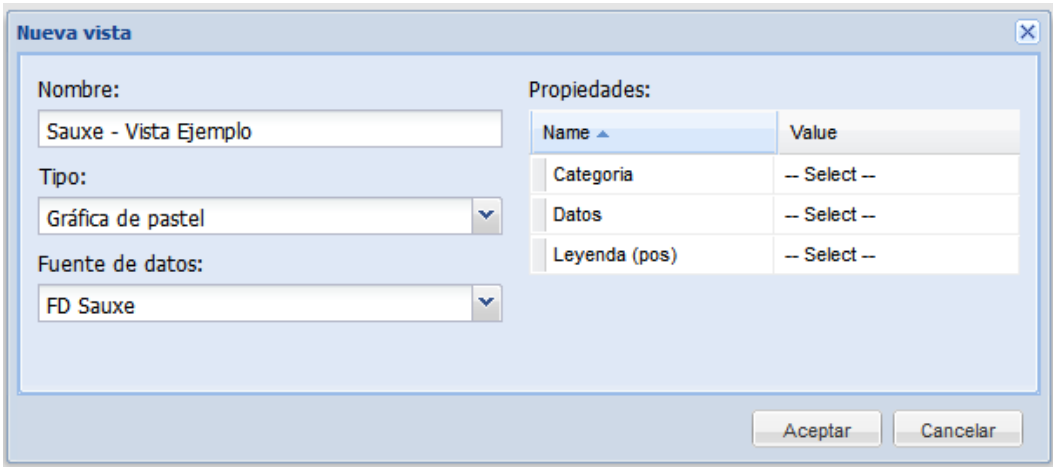


Figura 6: Interfaz Nueva Vista

Especificación del requisito: Eliminar vista.

Precondiciones	Debe existir al menos una vista.
Flujo de eventos	
Flujo básico	
1	Se selecciona el monitor que contiene la vista a eliminar
2	Se cierra la vista.
3	El sistema muestra un mensaje de confirmación para eliminar la vista.
4	Concluye el requisito.

Pos-condiciones		
1	Se eliminó la vista del monitor.	
Flujos alternativos		
NA		
Pos-condiciones		
1	NA	
Validaciones		
	NA	
Relaciones	**Requisitos Incluidos**	NA
	Extensiones	NA
Conceptos	**Vista**	Visibles en la interfaz: Nombre
Requisitos especiales	NA	
Asuntos pendientes	NA	

Prototipo elemental de interfaz gráfica de usuario.

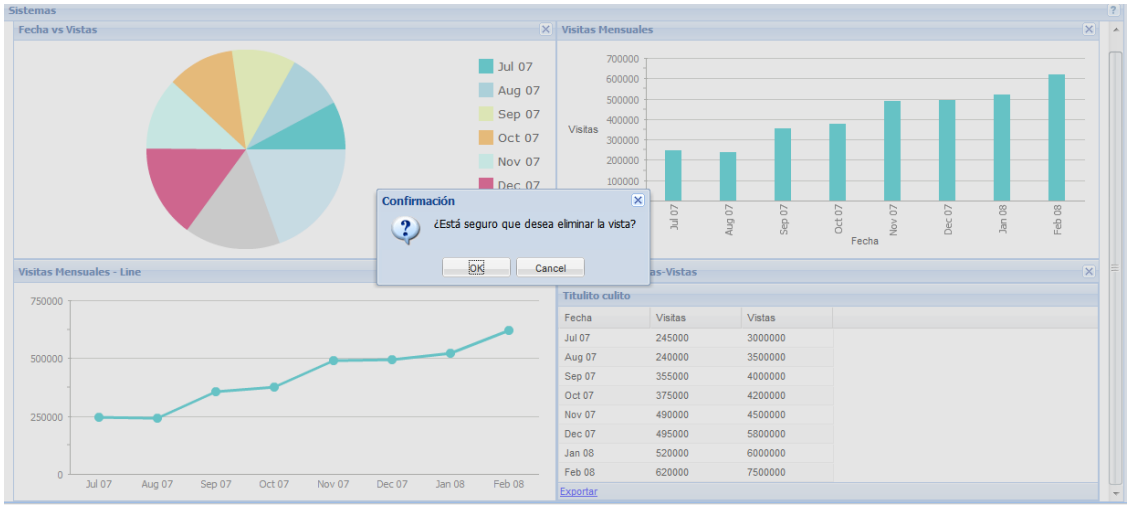


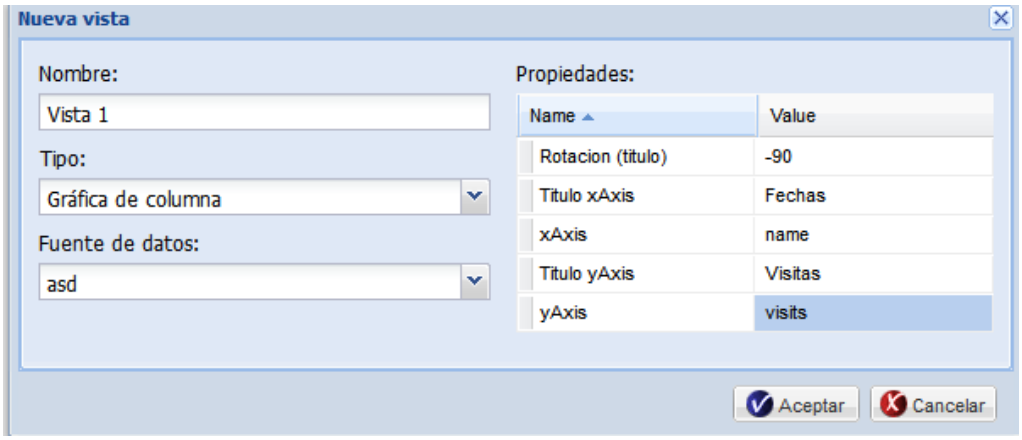
Figura 7: Interfaz Eliminar Vista 1.

Especificación del requisito: Definir el mapeo entre las fuentes de datos y las vistas.

Precondiciones	Debe existir una fuente de datos con estado en Lista Debe existir un monitor.
Flujo de eventos	
Flujo básico	
1 El sistema permite modificar las propiedades de la vista.	
2 Se introducen los datos de la vista. Gráfica de Pastel: – Categoría. – Valores. – Posición de la leyenda. Grafica de columna: - Rotación (título) - Título xAxis - xAxis - Título yAxis - yAxis Grafica de línea - Eje X - Eje Y Tabla - Se introduce el nombre de cada una de las columnas de la tabla.	
3 Concluye el requisito.	
Pos-condiciones	
1 Se configuraron las propiedades de la vista.	
Flujos alternativos	
Flujo alternativo *.a El usuario cancela la acción.	
1 Concluye el requisito.	
Validaciones	
NA	
Relaciones	**Requisitos** NA

	Incluidos	
	Extensiones	NA
Conceptos	**Fuente de Datos**	Visibles en la interfaz: Nombre Descripción Icono
Requisitos especiales	NA	
Asuntos pendientes	NA	

Prototipo elemental de interfaz gráfica de usuario



R3. Administrar monitores.

Especificación del requisito: Adicionar monitor.

Precondiciones	NA
Flujo de eventos	
Flujo básico	
1	Se introducen los datos: Nombre, Descripción, Icono.
2	El sistema comprueba que no exista un monitor con el mismo nombre.
3	Si los datos son correctos el sistema registra el monitor.
4	Concluye el requisito.

Pos-condiciones		
1	Se registró en el sistema un nuevo monitor.	
Flujos alternativos		
Flujo alternativo 2.a Existe un monitor con el mismo nombre.		
1	El sistema notifica que existe un monitor con el mismo nombre**.**	
2	Volver al paso 1	
Flujo alternativo 3.a Información incompleta.		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 3 del flujo básico.	
Flujo alternativo *.a El usuario cancela la acción.		
1	Concluye el requisito.	
Pos-condiciones		
1	NA	
Validaciones		
	NA	
Relaciones	**Requisitos Incluidos**	NA
	Extensiones	NA
Conceptos	**Monitor**	Visibles en la interfaz: Nombre Descripción Icono
Requisitos especiales	NA	
Asuntos pendientes	NA	

Prototipo elemental de interfaz gráfica de usuario

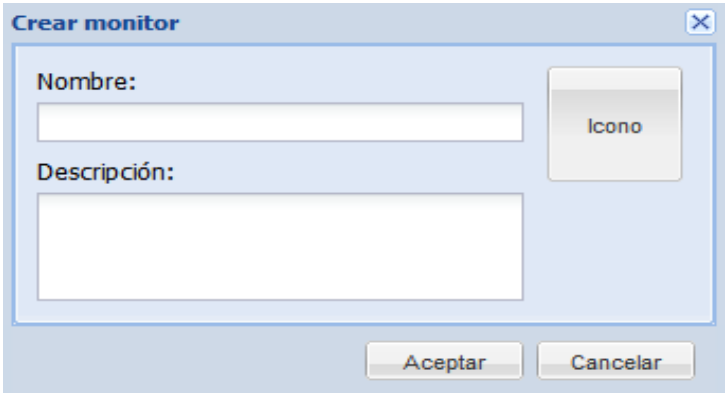


Figura 8: Interfaz Crear Monitor.

Especificación del requisito: Eliminar monitor.

Precondiciones		Se ha añadido al menos un monitor al sistema.
Flujo de eventos		
Flujo básico		
1	Se selecciona el monitor a eliminar.	
2	Se selecciona la opción eliminar.	
3	Se solicita confirmación para eliminar el monitor.	
4	Concluye el requisito.	
Pos-condiciones		
1	Se eliminó el monitor.	
Flujos alternativos		
Flujo alternativo *.a El usuario cancela la acción.		
1	Concluye el requisito.	
Pos-condiciones		
1	NA	
Validaciones		
	NA	
Relaciones	**Requisitos Incluidos**	NA
	Extensiones	NA

Conceptos	**Monitor**	Visibles en la interfaz: Icono Nombre
Requisitos especiales	NA	
Asuntos pendientes	NA	

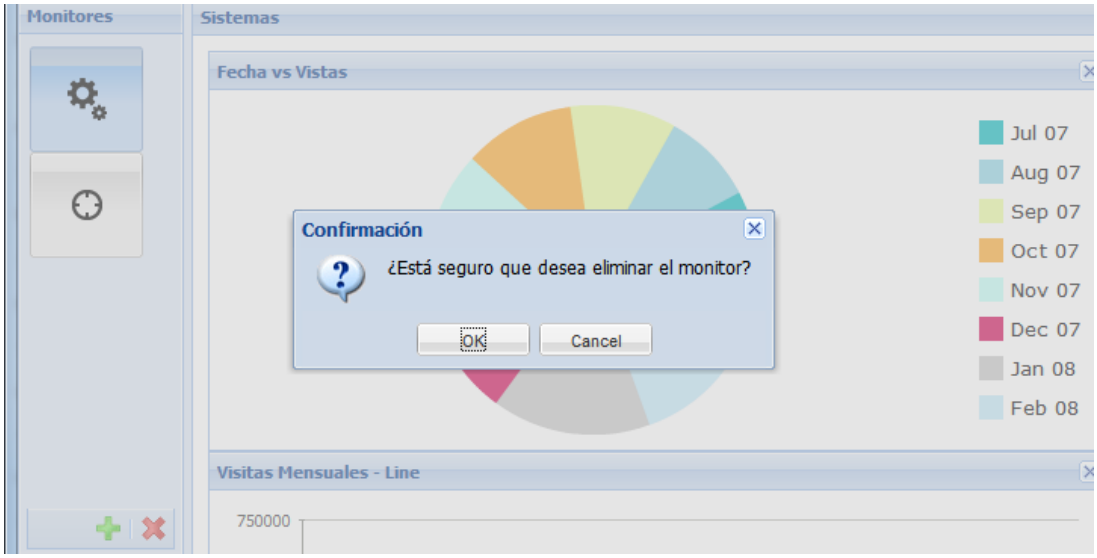


Figura 9: Interfaz Eliminar Monitor.

Especificación del requisito: Exportar tablas del monitor a hoja de cálculo.

Precondiciones	Debe existir al menos un monitor en el sistema y al menos una gráfica o tabla.
Flujo de eventos	
Flujo básico	
1	Se selecciona la opción de Exportar.
2	Se selecciona la ubicación donde se desea guardar.
3	Concluye el requisito.
Pos-condiciones	
1	Se exportó la tabla a hoja de cálculo.

Flujos alternativos		
Flujo alternativo		
1	NA	
Pos-condiciones		
1	NA	
Validaciones		
	NA	
Relaciones	**Requisitos Incluidos**	NA
	Extensiones	NA
Conceptos	NA	NA
Requisitos especiales	Se exportara al formato: hoja de cálculo de Microsoft Exel	
Asuntos pendientes	NA	

Prototipo elemental de interfaz gráfica de usuario.

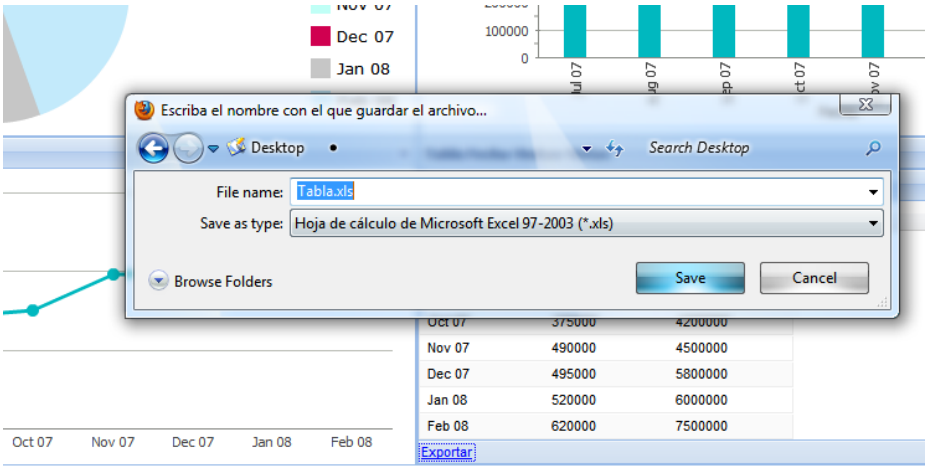


Figura 10: Interfaz Exportar tablas a hoja de cálculo.

	A	B	C
1	Tabla Fecha-Visitas-Vistas		
2	Name	Visits	Views
3	Jul 07	245000	3000000
4	Aug 07	240000	3500000
5	Sep 07	355000	4000000
6	Oct 07	375000	4200000
7	Nov 07	490000	4500000
8	Dec 07	495000	5800000
9	Jan 08	520000	6000000
10	Feb 08	620000	7500000

Figura 11: Salida del requisito Exportar tablas a hoja de cálculo.

2.5.2. REQUISITOS NO FUNCIONALES.

Eficiencia

1. El sistema para asegurar que los datos sean mostrador en tiempo real debe tener tiempos de respuesta en máximo de 500 milisegundos (esta cifra no incluye los retardos por concepto de tráfico de red).
2. El sistema debe soportar hasta 100 conexiones simultáneas.

Restricciones de diseño

1. El sistema será un módulo para el marco de trabajo Sauxe.
2. El sistema se implementará usando el lenguaje PHP.
3. El sistema usará el Framework de Presentación ExtJS en su versión 3.4.0.

Seguridad

1. Autenticación (Contraseña de acceso).
2. Verificación sobre las acciones irreversibles (eliminaciones).

Software

3. El dashboard requiere que en el entorno de ejecución se encuentren instaladas y configuradas las siguientes aplicaciones:
4. Servidor Web Apache versión 2.0 con soporte para PHP 5.2.
5. Mozilla Firefox versión superior a la 3.6.
6. Servidor Jabber sobre el protocolo XMPP.

Interfaces Hardware

En aras del mejor comportamiento del sistema se requiere que en el entorno de ejecución y estaciones de trabajo se encuentren disponibles las siguientes características:

Servidor.

1. Procesador: 3.00 GHz.
2. RAM: 1 Gb.
3. Disco duro: 80 Gb.
4. Tarjeta de red.

Cliente.

1. Procesador: 1.4 GHz.
2. RAM: 256 Mb
3. Disco duro: 80 Gb.
4. Tarjeta de red.

2.6. MODELO DE DISEÑO

El Modelo de diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es utilizado como entrada esencial en las actividades relacionadas a la implementación. El Modelo de Diseño puede contener: diagramas, clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones y atributos. (42)

2.6.1. PATRONES UTILIZADOS.

Patrón arquitectónico Modelo Vista Controlador

El patrón de arquitectura conocido como Modelo-Vista-Controlador (MVC), separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario; es decir separa en tres capas diferentes los datos de una aplicación, la interfaz de usuario, y la lógica de control:

Modelo: Esta capa administra el comportamiento y los datos del dominio de la aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Esta capa maneja la visualización de la información, es decir que presenta el modelo en un formato adecuado para interactuar, que usualmente es la interfaz de usuario.

Controlador: Esta capa controla el flujo de datos entre la vista y el modelo; es decir que responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista, tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. (43)

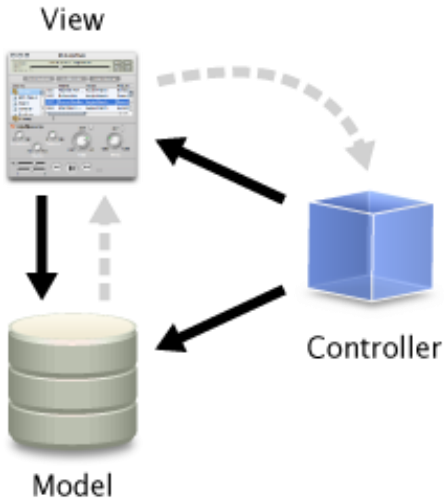


Figura 12: Modelo-Vista-Controlador.

2.6.2. PATRONES DE DISEÑO

Patrones GoF (Gang of Four).

- **Observer (Observador):** clasifica como patrón de comportamiento a nivel de objeto, el problema que lo motiva es la existencia de diferentes objetos desacoplados que deben mantenerse actualizados del estado de otro objeto, o de los determinados sucesos que ocurren en él, su propósito es garantizar que los interesados en el estado del objeto observado sean notificados convenientemente. (23)

  El observador es ampliamente utilizado en el diseño de componentes de interface de usuario de la capa de presentación. Define una relación de un objeto a muchos objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este cambio a todos los otros objetos. Su uso se evidencia en la clase MessageBuss.js.

2.6.3. DIAGRAMA DE CLASES DEL DISEÑO.

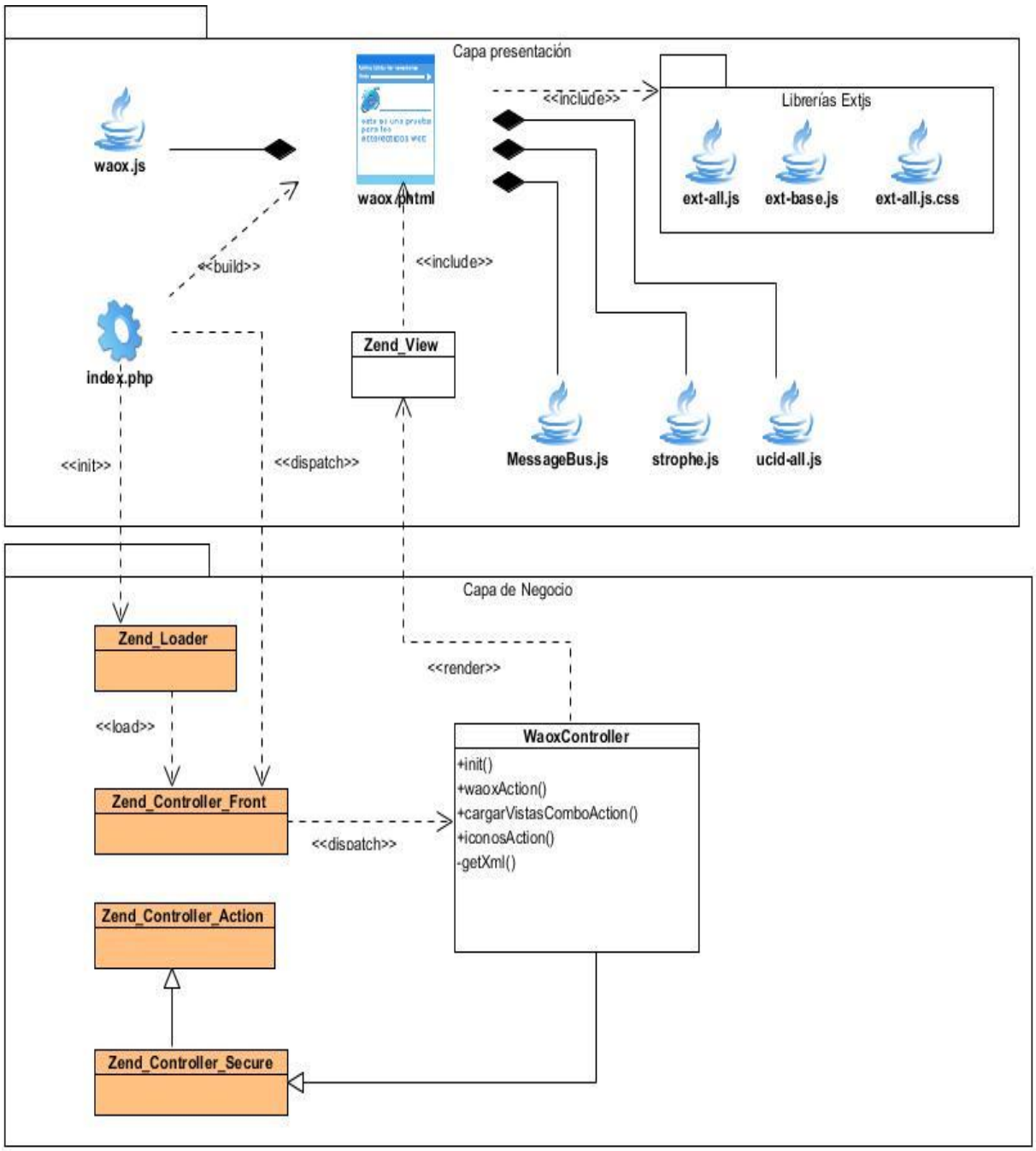


Figura 13: Diagrama de Clases del Diseño.

Descripción de clases significativas para el diseño

Nombre: WaoXController	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
Init()	Constructor de la clase.
cargarVistasComboAction()	Funcionalidad encargada de cargar los datos para llenar el combo box que hace referencia a los Tipos de Vistas.
iconosAction()	Funcionalidad encargada de listar las imágenes para confeccionar el selector de iconos del monitor.
getXml()	Funcionalidad encargada de cargar el fichero de configuración main.xml, el cual posee los Tipos de Vistas.

2.6.4. DIAGRAMA DE COMPONENTES

En el diagrama de componentes se muestran los elementos de diseño de un sistema de software. Permite visualizar con más facilidad la estructura general del sistema y el comportamiento de los servicios que estos componentes proporcionan y utilizan a través de las interfaces. (44)

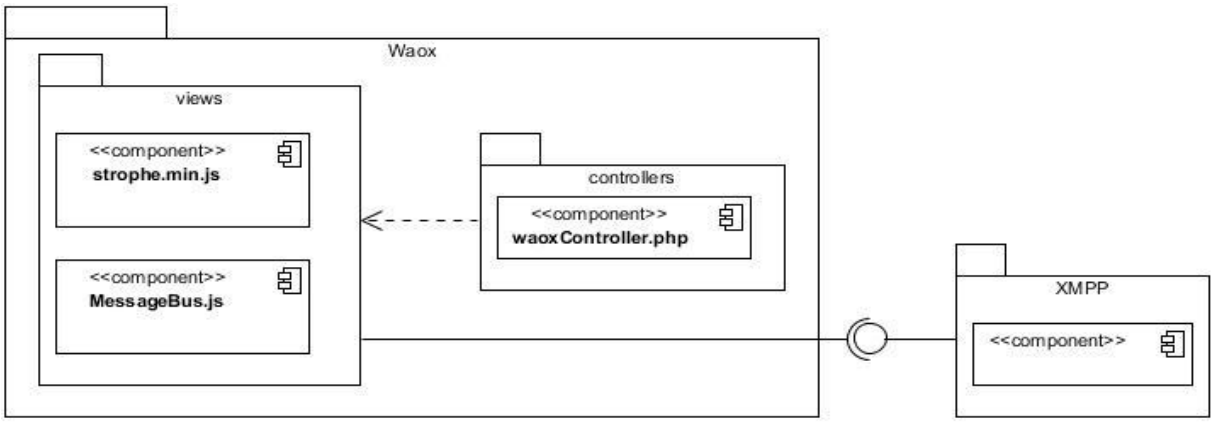


Figura 14: Diagrama de Componentes.

2.6.5. DIAGRAMA DE DESPLIEGUE

El Diagrama de Despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos, muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento.

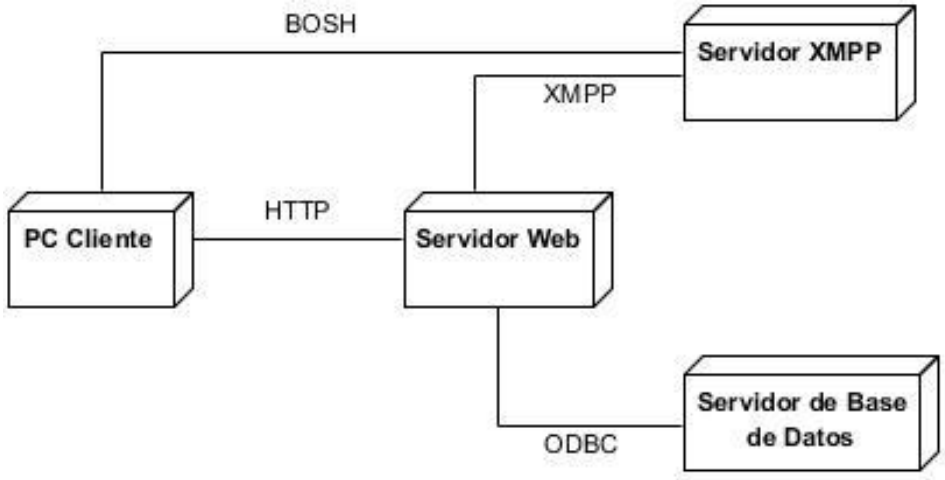


Figura 15: Diagrama de Despliegue.

2.7. CONCLUSIONES

Con el desarrollo de este capítulo se conocieron los procesos objeto de automatización, lo que proporcionó una temprana idea de la complejidad de la solución. Apoyados del modelo conceptual con los conceptos más importantes y de los requisitos funcionales se logró tener una idea más detallada de lo que se quiere implementar, los requisitos no funcionales permitieron garantizar una ejecución eficaz de la aplicación.

Teniendo en cuenta la arquitectura del sistema se conformó la arquitectura base que rige el diseño y el diagrama de clases del diseño. La descripción de clases importantes permitió tener una visión para la puesta en práctica de patrones. Se establecieron las bases para poder pasar a la implementación de la solución.

CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1. INTRODUCCIÓN

La dificultad para construir sistemas de software multiplica la probabilidad de que persistan errores aún después de haberse finalizado. En el desarrollo del software, las posibilidades de error son innumerables. Pueden darse por una mala especificación de los requisitos funcionales, uso indebido de las estructuras de datos, errores al enlazar módulos, etc. Es imposible asegurar que un software se encuentre completamente libre de errores; sin embargo, existen formas y métodos para acercarse lo más posible a un resultado óptimo. En este capítulo se realiza la validación de la solución propuesta.

3.2. PRUEBAS DE SOFTWARE

Las pruebas de software son el conjunto de técnicas que permiten determinar la calidad de un producto. Estas se integran dentro de las diferentes fases del Ciclo del Vida del software dentro de la Ingeniería. Así se ejecuta un programa y mediante técnicas experimentales se trata de descubrir que errores presenta. La calidad de un sistema informático es algo subjetivo que depende del contexto y del objeto que se pretenda conseguir. Para determinar dicho nivel de calidad se deben efectuar medidas o pruebas que permitan comprobar el grado de cumplimiento con respecto a las especificaciones iniciales del sistema.

3.2.1. OBJETIVOS DE LAS PRUEBAS.

El objetivo de las pruebas de un programa es el de detectar todo posible mal funcionamiento, un error puede ser costoso de reparar mientras más se avanza en las etapas del ciclo de vida del software. Dentro de los objetivos fundamentales que se persiguen al aplicarles pruebas a un software se encuentran los siguientes:

- Brindar un mayor nivel de confiabilidad en los productos que se van generando.
- Detectar fallas o errores.
- Aumentar la calidad del producto final.

Si las pruebas que se llevan a cabo tienen éxito se descubrirán errores en el software, dándole mayor fiabilidad. Es importante tener en cuenta una frase de Pressman: *"La prueba no puede asegurar la ausencia de defectos; solo puede demostrar que existen defectos en el software".* (32)

3.3. MODELOS DE PRUEBAS

Existen 2 enfoques principales de prueba:

- El enfoque estructural o de caja blanca: que se basa en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- El enfoque funcional o de caja negra: que realiza pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.

3.3.1. PRUEBAS DE CAJA BLANCA O ESTRUCTURALES.

A este tipo de técnicas se le conoce también como Técnicas de Caja Transparente o de Cristal. Este método se centra en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa examinando su lógica interna sin considerar los aspectos de rendimiento. El objetivo de la técnica es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa.

A continuación se citan algunas de las técnicas de prueba de Caja Blanca:

- Prueba de Condición.
- Prueba de Flujo de Datos.
- Prueba de Bucles.
- Prueba del Camino Básico.

Dentro de la prueba de caja blanca, la técnica que se utilizó fue la de prueba del camino básico. Esta prueba permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. (32)

Para aplicar la técnica del camino básico se debe introducir la notación para la representación del flujo de control, este puede representarse por un Grafo de Flujo en el cual:

1. Cada nodo del grafo corresponde a una o más sentencias de código fuente.
2. Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo.
3. Se calcula la complejidad ciclomática del grafo.

Un grafo de flujo está formado por 3 componentes fundamentales que ayudan a su elaboración y comprensión, estos brindan información para confirmar que el trabajo se está haciendo adecuadamente.

Componentes del grafo de flujo:

- Nodo: son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.
- Aristas: son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.
- Regiones: son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

Para realizar la técnica de prueba de caja blanca, específicamente la prueba del camino básico es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumera las sentencias de código del procedimiento realizado sobre el método cargarVistasComboAction ().

```
function cargarVistasComboAction ()
{
    $className = 'ComboBoxVistas';          //1
    $xml = $this->getXml ()->$className;    //1
    $result = array ();                     //1

    foreach ($xml->children () as $child) { //2
        if ((string) $child ['enabled'] === 'true') //3
            $result[] = array ('className' => (string) $child->getName (),  //4
                               'name'      => (string) $child ['name']);    //4


    }   //5

    echo json_encode(array ('data' => $result)); //6
}
```

Figura 16: Sentencias de código.

Después de este paso, es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones, en ese caso:

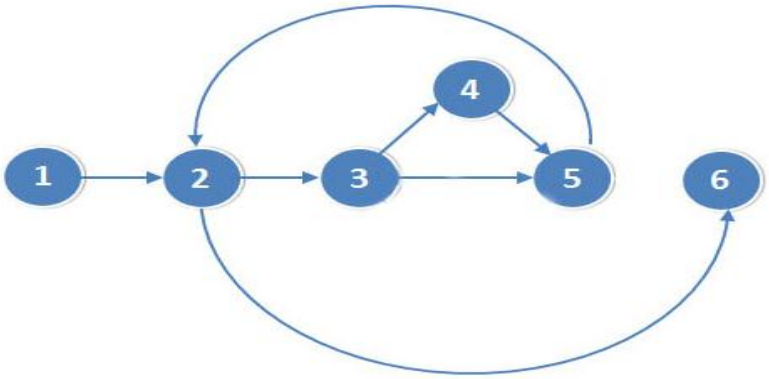


Figura 17: Grafo de flujo.

Cálculo de la complejidad ciclomática.

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un

límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. (32)

Para efectuar el cálculo de la complejidad ciclomática del código es necesario tener varios parámetros como son la cantidad total de aristas del grafo 7, cantidad total de nodos 6, cantidad total de nodos predicados 2 y la cantidad total de regiones 3, para las siguientes fórmulas:

1. **V (G) = (A - N) + 2**

Siendo “A” la cantidad total de aristas y “N” la cantidad de nodos.

V (G) = (7 - 6) + 2

V (G) = 3

2. **V (G) = P + 1**

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 2 + 1 = 3

3. **V (G) = R**

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.

V (G) = 3

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, dando como resultado 3, lo que indica que existen 3 posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

- **Camino básico #1:** 1 – 2 – 3 – 4 – 5– 2 – 6
- **Camino básico #2:** 1 – 2 – 3 – 5 – 2 – 6
- **Camino básico #3:** 1 – 2 – 6

Para cada camino se realiza un caso de prueba.

1. Caso de prueba para el camino básico #1.

Descripción: Los datos de entrada se obtienen a través del método getXml(), que le da valor a la variable xml.

Condición de ejecución: Los datos no deben ser nulos.

Obtenido: Un arreglo con los tipos de vistas en el índice “data”. Array ('data' => $result)

Resultados esperados: Un arreglo con los tipos de vistas en el índice “data”.

2. Caso de prueba para el camino básico #2.

Descripción: Los datos de entrada se obtienen a través del método getXml(), que le da valor a la variable xml.

Condición de ejecución: Los datos no deben ser nulos.

Obtenido: Un arreglo vacío.

Resultados esperados: Un arreglo vacío.

3. Caso de prueba para el camino básico #3.

Descripción: Los datos de entrada se obtienen a través del método getXml(), que le da valor a la variable xml.

Condición de ejecución: Los datos no deben ser nulos.

Obtenido: Un arreglo vacío.

Resultados esperados: Un arreglo vacío.

Luego de aplicar los distintos casos de pruebas, se pudo comprobar que el flujo de trabajo de la función es correcto ya que cumple con las condiciones necesarias que se habían planteado.

3.3.3. PRUEBAS DE CAJA NEGRA O DE COMPORTAMIENTO.

Las pruebas de caja negra también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

Los casos de prueba demuestran que:

- Las funciones del software son operativas.
- Las entradas se aceptan de la forma adecuada produciendo el resultado correcto.
- La integridad de la información externa (por ejemplo archivos de datos) se mantiene.

Estas pruebas permiten encontrar errores tales como:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores de estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación. (45)

Algunas técnicas utilizadas en las pruebas de caja negra son:

- **Partición de equivalencia:** Técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.
- **Análisis de valores límite:** La experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen. Las condiciones límite son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica lleva a elegir los casos de prueba que ejerciten los valores límite.
- **Grafos de causa-efecto:** Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Se utilizará en particular la técnica de partición de equivalencia en la prueba de caja negra que se aplicará. Para la aplicación de este tipo de pruebas se diseñaron un conjunto de escenarios de pruebas con el objetivo de evaluar el cumplimiento de los requisitos del sistema. A continuación se muestran los escenarios empleados.

Tabla III: Escenarios - Requisito Adicionar Fuente de Datos.

Nombre del requisito.	**Descripción general.**	**Escenarios de pruebas.**	**Flujo del escenario.**
1: Adicionar fuente de datos.	El sistema debe permitir adicionar fuentes de datos.	EP 1.1: Adicionar fuente de datos introduciendo datos válidos.	– Se introducen los datos de la fuente de datos correctamente. – Se presiona el botón **Aceptar.** – Se muestra el administrador de fuentes.
		EP 1.2: Adicionar fuente de datos introduciendo datos inválidos.	- El sistema no permite introducir datos inválidos.
		EP 1.3: Adicionar fuente de datos introduciendo un código existente.	- Se introducen los datos del dispositivo con un código ya existente. - Se presiona el botón **Aceptar.** - Se muestra un

			mensaje informando del error.
		EP 1.4: Adicionar fuente de datos introduciendo un nombre existente.	- Se introducen los datos del dispositivo con un nombre ya existente. - Se presiona el botón **Aceptar.** Se muestra un mensaje informando del error.
		EP 1.5: Adicionar fuente de datos dejando campos vacíos.	- Se introducen los datos de la fuente de datos dejando algún campo en blanco. - Se muestra un mensaje informando del error.
		EP 1.6: Cancelar	- Se introducen o no los datos de la fuente de datos. - Se presiona el botón **Cancelar**.

Tabla IV: Descripción de las Variables – Requisito Adicionar Fuente de Datos.

No	Nombre de campo	Tipo	Válido	Inválido	Inválido
1	Nombre.	Campo de texto.	Letras y espacios.	Números, caracteres especiales.	Vacío.
2	Código	Campo de texto.	Letras y números.	Caracteres especiales, espacios.	Vacío.
3	Descripción	Campo de texto.	Letras, números, caracteres especiales.	NA	NA

Tabla V: Juego de Datos – Requisito Adicionar Fuente de Datos.

Id del escenario	Escenario	Nombre	Código	Descripción	Respuesta del sistema	Resultado de la prueba
EP 1.1	Adicionar fuente de datos introduciendo datos válidos.	V(Adicionar fuente)	V(00Rf2)	V (Prueba RF 1)	Se muestra la ventana Administrador de fuentes. Con la nueva fuente adicionada.	Se mostró la nueva fuente en el administrador de fuentes.
EP 1.2	Adicionar fuente de datos introduciendo datos	V(*n@mbre 1_)	V(a$1-r C)	NA	El sistema no permite la escritura de caracteres inválidos.	Se muestran los campos de texto en rojo.

	inválidos.					
EP 1.3	Adicionar fuente de datos introduciendo un código existente	V(nombreUNO)	V(001)	V(Prueba RF 1)	El sistema muestra un mensaje que diga. "El código de la fuente de datos ya existe"	Se mostró el mensaje: "El código de la fuente de datos ya existe"
EP 1.4	Adicionar fuente de datos introduciendo un nombre existente.	V(Nombre UNO)	V(002)	V(Prueba RF 1)	El sistema muestra un mensaje que diga: "El nombre de la fuente de datos ya existe"	Mostró un mensaje. "El nombre de la fuente de datos ya existe"
EP 1.5	Adicionar fuente de datos dejando campos vacíos.	Vacío	Vacío	Vacío	El sistema muestra un mensaje que diga: "Por favor verifique los campos incorrectos o vacíos"	Mostró el mensaje: "Por favor verifique los campos incorrectos o vacíos"
		V(Nombre UNO)	Vacío	V(Prueba RF 1)		

EP 1.6	Cancelar	V(Nombre UNO)	NA	NA	Se cancela la acción.	Se canceló correctamente la acción.

Resultados de las pruebas de caja negra.

Para la realización de las pruebas de caja negra al sistema fueron analizados los casos de pruebas, de los requisitos del sistema. Para esto se realizó una descripción de diferentes escenarios de prueba, los cuales fueron entre 3 y 6 escenarios a probar, los mismos se probaron entre 1 y 4 juegos de datos. Atendiendo el comportamiento del sistema ante diferentes situaciones (entradas válidas y no válidas). Los errores que arrojaron el resultado de esta prueba fueron las faltas de ortografía en las interfaces y los mensajes de información o de errores, los cuales fueron corregidos para corroborar que se cumplían con los resultados esperados.

En la siguiente grafica se muestra la cantidad y tipos de errores encontrados en cada iteración realizada.

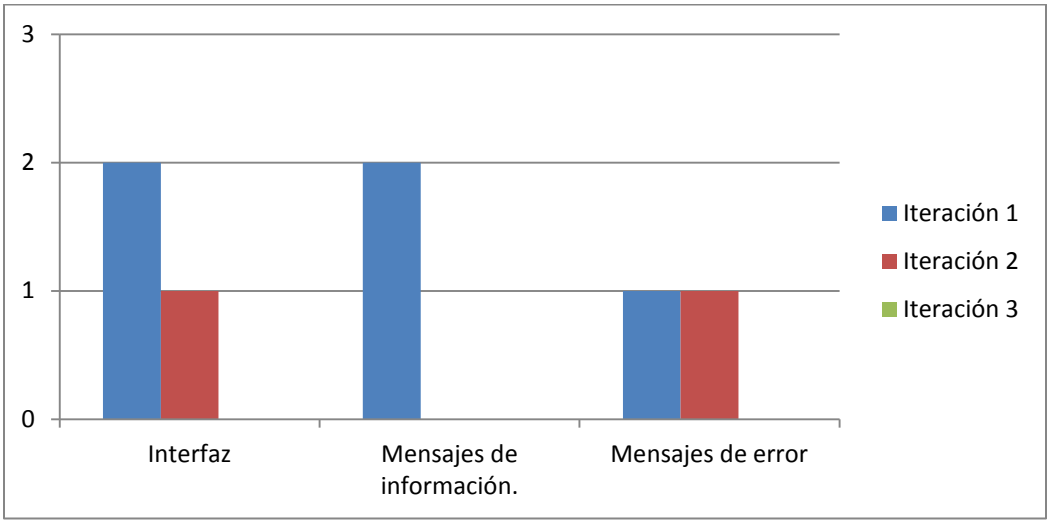


Figura 18: Resultados Prueba de Caja Negra

3.4. CONCLUSIONES

Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la fase de pruebas. Esta fase del desarrollo de un software es la que mayor cantidad de tiempo y de esfuerzo requiere, se estima que la mitad del esfuerzo de desarrollo de un programa tanto en tiempo como en gastos se invierte en esta. Esta fase añade valor al producto, todos los programas poseen errores y la fase de pruebas los descubre, siendo este el valor que le añade. Mientras más errores se encuentren al software en esta fase mejor.

En el presente capítulo se efectuaron las pruebas de software necesarias para lograr la calidad requerida de la aplicación propuesta. Fueron validadas las funcionalidades implementadas a través de las pruebas de caja negra, mostrando cómo respondían adecuadamente a los requisitos funcionales y garantizando la satisfacción plena de las necesidades reales de los usuarios y demandas del cliente. De igual forma, se aplicaron pruebas de caja blanca para efectuar las revisiones al código, las cuales dieron resultados positivos. Desde el punto de vista funcional la aplicación cumple con los requerimientos capturados y especificados en las primeras etapas de desarrollo.

CONCLUSIONES

En el presente trabajo se desarrolló una revisión bibliográfica de los distintos conceptos y aspectos necesarios para entender el objetivo fundamental del trabajo, un estudio de sistemas similares existentes y las tecnologías abiertas que mejor sustancien la integración de una plataforma de monitoreo en tiempo real con el marco de trabajo Sauxe. Esta revisión arrojó como resultado la posibilidad de creación de una nueva plataforma que bajo la misma tecnología que Sauxe potencie la visualización del estado de un sistema informático en un tiempo real.

A través de la arquitectura propuesta por el marco de trabajo Sauxe y la metodología de desarrollo del CEIGE se obtuvo el modelo de dominio y el análisis de la solución propuesta. Bajo un estilo arquitectónico en capas sobre la base del patrón MVC definido por el marco de trabajo Zend, se sostuvo el diseño y la implementación de la solución.

Las pruebas estructurales y funcionales realizadas a la aplicación validaron el correcto funcionamiento de la misma. Como resultado final se obtuvo la implementación de una aplicación integrable al marco de trabajo Sauxe que permite la visualización gráfica y en tiempo real de la información generada por las diferentes aplicaciones desarrolladas en el mismo, mejorando el proceso de toma de decisiones y la agilidad en el análisis de información.

RECOMENDACIONES

Ningún sistema es estático en el tiempo debido a que la evolución del negocio y el desarrollo de las tecnologías así lo exigen, es por ello que se recomienda:

- El resultado propuesto en la investigación sea implantado en las soluciones Sistema de Gestión Integral Cedrux, Sistema de Mantenimiento Vehicular y Sistema de Administración de Relaciones con el Cliente, así como nuevas aplicaciones web de gestión que se desarrollen sobre el Marco de Trabajo Sauxe.

- Evaluar el funcionamiento de la herramienta integrada a otras aplicaciones de tecnologías diferentes.

- Incluir una funcionalidad que permita guardar la persistencia sobre la configuración del usuario, mediante el uso de un archivo de configuración o una base de datos.

BIBLIOGRAFÍA

1. Marketing Electrónico. *CRM y ERP.* [En línea] [Citado el: 20 de Noviembre de 2011.] http://www.marketingelectronico.com/blog/que-diferencia-hay-entre-erp-y-crm/.

2. **Canney Restrepo, Edward.** TodoBI. [En línea] 23 de Agosto de 2007. [Citado el: 9 de Novienmbre de 2011.] http://todobi.blogspot.com/2007/08/la-respuesta-esta-en-los-dashboards.html.

3. **Few, Stephen.** *Common Pitfalls in Dashboard Design.* 2006.

4. **Malik, Shadan.** *Enterprise Dashboard:: Design and Best Practices for IT.* Hoboken, New Jersey. : John Wiley & Sons, Inc., 2005.

5. **Kirkpatrick, Marshall.** ReadWriteWeb. *Explaining the Real-Time Web in 100 Words or Less.* [En línea] 22 de Septiembre de 2009. [Citado el: 1 de Diciembre de 2011.] http://www.readwriteweb.com/archives/explaining_the_real-time_web_in_100_words_or_less.php.

6. **DZ, Mark P.** Dashboard Zone. [En línea] [Citado el: 6 de diciembre de 2011.] http://www.dashboardzone.com/what-is-a-dashboard.

7. **Tuck, Allene, Pyne, Sandra y Ashby, Michael F.** *Oxford dictionary of computing for learners of English.* s.l. : Oxford: Oxford University Press. 0194314413.

8. **Young, S.J.** *Real Time Languages: Desing and Development.* 1982.

9. **Burns, Alan y Wellings, Andy.** *Real-time systems and programming languages.* University of York : Addison Wesley, 1996.

10. **Gutiérrez, Javier J.** ¿Qué es un framework web? *Departamento de Lenguajes y Sistemas Informaticos.* [En línea] 2009. [Citado el: 5 de diciembre de 2011.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

11. **Pentaho.** Pentaho. Powerful Analytics Made Easy. [En línea] [Citado el: 20 de enero de 2012.] http://community.pentaho.com.

12. **OpenReports.** *OpenReports Administration Guide.* [Documento]

13. IBM. *IBM Cognos 8 Platform.* [En línea] [Citado el: 1 de Diciembre de 2011.] http://www-01.ibm.com/software/data/cognos/products/cognos-8-platform/.

14. IBM. *International Business Machines Corporation.* [En línea] [Citado el: 6 de Diciembre de 2011.] http://www-01.ibm.com/software/analytics/cognos-8-go/dashboard/.

15. **IBM.** *IBM Cognos 8 Go! Dashboard.* [Documento] s.l. : IBM Corporation, 2006.

16. **SAS.** *SAS(R) BI Dashboard 4.31: User's Guide.* [En línea] [Citado el: 18 de Diciembre de 2011.] http://support.sas.com/documentation/cdl/en/bidbrdug/64524/HTML/default/viewer.htm#n1t3l2n9us6eyyn1nm98l2wfedh7.htm.

17. **Builders, Information.** *WebFocus 8.* [Documento] New York : s.n., 2010. DN7506431.0410.

18. **InetSoft.** InetSoft. [En línea] [Citado el: 21 de Enero de 2012.] http://www.inetsoft.com/products/free_dashboard_software/.

19. **Scavone, Federico.** *XMPP.* s.l. : Universidad Católica Nuestra Señora de la Asunción. Facultad de Ciencias y Tenología., 2010.

20. **Saint-Andre, Peter, Smith, Kevin y Tronçon, Remko.** *XMPP: The Definitive Guide.* s.l. : O'Reilly Media, Inc, 2009. ISBN: 978-0-596-52126-4.

21. **Paterson, Ian.** XMPP Over BOSH. *XMPP.* [En línea] 27 de diciembre de 2011. http://xmpp.org/extensions/xep-0206.html..

22. **Moffitt, Jack.** *Professional XMPP Programming with JavaScript and jQuery.* Indianapolis, : Wiley Publishing, Inc, 2010. ISBN: 978-0-470-54071-8.

23. **Gómez Baryolo, Oinier, Morejón Borbón, Yoandry y García Tejo, Darien.** *ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.* Universidad de las Ciencias Informaticas. Ciudad de La Habana : s.n., 2009.

24. Zend Framework. [En línea] Zend Technologies, 2011. [Citado el: 7 de diciembre de 2011.] http://framework.zend.com/manual/en/introduction.overview.html.

25. **Corzo, Giancarlo.** Desarrollo en Web. *Blog sobre desarrollo de aplicaciones web en Java, Python y JavaScript.* [En línea] 22 de Octubre de 2008. [Citado el: 7 de Diciembre de 2011.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

26. **Mata, Manel Pérez.** TecnoRetales. [En línea] 3 de Julio de 2009. [Citado el: 7 de Diciembre de 2011.] http://www.tecnoretales.com/programacion/que-es-doctrine-orm/.

27. **PHP Documentation Group.** PHP: Hypertext Pre-processor. [En línea] 2 de Diciembre de 2011. [Citado el: 7 de Diciembre de 2011.] http://www.php.net/manual/es/index.php.

28. **Pérez, Javier Eguíluz.** *Introducción a Javascript.* 2009.

29. **Benz, Brian y Durant, John R.** *XML Programming Bible.* s.l. : Wiley Publishing. ISBN: 0-7645-3829-2.

30. **Pérez, Javier Eguíluz.** Introduccion a CSS. *Libros Web.* [En línea] 2008. http://www.librosweb.es/css.

31. The JSON Specification. [En línea] [Citado el: 20 de abril de 2012.] http://json.org.

32. **Pressman, Roger S.** *Ingenieria de Software. Un enfoque Práctico 5ta Edición.* Madrid : Concepción Femández Madrid, 2002.

33. NetBeans. *Oracle Corporation.* [En línea] [Citado el: 10 de Diciembre de 2011.] http://netbeans.org/index_es.html.

34. **Loeliger, Jon.** *Version control with Git.* s.l. : O'Reilly, 2009. ISBN: 978-0-596-52012-0.

35. **Copyleft.** Covetel. *Cooperativa Venezolana de Tecnologías Libres R.S.* [En línea] [Citado el: 10 de Diciembre de 2011.] http://www.covetel.com.ve/productos/seguimiento.html.

36. **Mateu, Carles.** *Desarrollo de aplicaciones web.* Barcelona : Eureca Media, SL, 2007. ISBN: 84-9788-118-4.

37. **Ford, A.** *Apache 2 pocket reference.* s.l. : O'Reilly Media, Apache 2 pocket reference.

38. **Drake, J y Worsley, J.** *Practical PostgreSQL.* s.l. : O'Reilly Media, Inc., 2011.

39. OpenSuse. [En línea] Novell, Inc, 12 de Octubre de 2009. [Citado el: 10 de Diciembre de 2011.] http://es.opensuse.org/Postgresql.

40. **Larman, Craig.** *UML y Patrones: Introducción al análisis oreientado a objetos.* s.l. : Pearson.

41. **Synergix.** Tecnología y Synergix. *Tecnología y Synergix.* [En línea] [Citado el: 27 de Febrero de 2011.] http://synergix.wordpress.com/2008/07/07/requisito-funcional-y-no-funcional/.

42. MeRinde. [En línea] [Citado el: 10 de abril de 2012.] http://merinde.net/index.php?option=com_content&task=view&id=94&Itemid=296.

43. **Marquina, Ernesto y Parra, Jose David.** *Guía de Patrones, Practicas y Arquitectura .NET.* [Documento] 2008.

44. **Microsoft.** MSDN. [En línea] [Citado el: 20 de Abril de 2012.] http://msdn.microsoft.com/.

45. **Huenca, Daniel.** Herramientas de Prueba de Software. [En línea] [Citado el: 24 de Mayo de 2012.] http://herrorsoft.zxq.net/.

GLOSARIO DE TÉRMINOS

Base de datos: Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Caso de Prueba: Conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio.

Cliente: Persona, organización o grupo de personas que encarga la construcción de un sistema, ya sea empezando desde cero, o mediante un refinamiento de versiones sucesivas.

Componente: Es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.

Mapeo: Relación entre una clase y su almacenamiento persistente (p.ej. una tabla de la BD), y entre los atributos del objeto y los campos (columnas) de un registro.

Métrica: En el campo de la ingeniería del software una métrica es cualquier medida o conjunto de medidas destinadas a conocer o estimar el tamaño u otra característica de un software o un sistema de información, generalmente para realizar comparativas o para la planificación de proyectos de desarrollo.

Metodología Ágil: Se basan en retrasar las decisiones y la planificación adaptativa; permitiendo potenciar aún más el desarrollo de software a gran escala. Estas metodologías ponen en relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan. Como parte de ésta aparecen Extreme Programing (XP), SCRUM y Ágil Unified Process (AUP).

Patrones: Normas de comportamiento, características que identifican una situación. En informática un patrón es una solución a un problema de diseño no trivial que es efectiva y reutilizable. Es posible aplicar a diferentes problemas de diseño en distintas circunstancias.

Patrones de diseño: (Design patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño.

Programación Orientada a Objetos (OOP por sus siglas en inglés de Object Oriented Programming) como paradigma, "es una forma de pensar, una filosofía, de la cual surge una cultura nueva que incorpora técnicas y metodologías diferentes. Pero estas técnicas y metodologías, y la cultura misma, provienen del paradigma, no lo hacen. La OOP como paradigma es una postura ontológica: el universo computacional está poblado por objetos, cada uno responsabilizándose por sí mismo, y comunicándose con los demás por medio de mensajes".

Pruebas de Software: Son los procesos que permiten verificar y revelar la calidad de un producto software.

Reporte: Un reporte es un informe o una noticia. Este tipo de documento (que puede ser impreso, digital, audiovisual, etc.) pretende transmitir una información, aunque puede tener diversos objetivos. En el ámbito de la informática, los reportes son informes que organizan y exhiben la información contenida en una base de datos. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios.

Tiempo de Respuesta: El tiempo de un sistema genérico, o unidad funcional necesario para reaccionar ante una entrada dada.